3D-Rekonstruktion von CAD-Objekten und momentenbasiertes 3D-Fitting

Dissertation

zur Erlangung des akademischen Grades

doctor rerum naturalium (Dr. rer. nat.)

vorgelegt dem Rat der

Fakultät für Mathematik und Informatik

der Friedrich-Schiller-Universität Jena

von Dipl.-Math. Frank Ditrich

geb. am 23.11.1971 in Gera

Inhaltsverzeichnis

Ich möchte mich an dieser Stelle bei allen bedanken, die zum Gelingen der Arbeit beigetragen haben.

Mein besonderer Dank gilt meinem Betreuer Dr. Herbert Süße für die ständige Bereitschaft offene Fragen und Probleme jederzeit klären zu können, für viele interessante und hilfreiche Diskussionen und für das angenehme und konstruktive Arbeitsumfeld. Dafür danke ich ebenfalls Prof. Dr. Klaus Voss, Prof. Dr. Joachim Denzler sowie allen Mitarbeitern des Lehrstuhls für Digitale Bildverarbeitung.

Ebenso danke ich hier der Geschäftsleitung und allen Mitarbeitern der DAKO GmbH Jena, besonders ihrem Geschäftsführer Joachim Becker. Meine Tätigkeit dort lieferte den Anstoß für das Thema der Dissertation, und die DAKO GmbH stellte den zeitlichen und finanziellen Rahmen für die Anfertigung der Arbeit zur Verfügung. Bei auftretenden Schwierigkeiten wurde ich stets schnell und problemlos unterstützt.

Desweiteren danke ich Prof. Dr. Hans-Dietrich Hecker für sein Engagement bei der Schaffung der Rahmenbedingungen für meine Arbeit seitens der Universität.

Nicht zuletzt waren auch die Unterstützung durch meine Familie und meine Freunde und das aufgebrachte Verständnis für meine oft sehr knapp bemessene Zeit gerade in manchen nicht ganzen einfachen Zeiten sehr wertvoll und wichtig für mich.

Bedanken möchte ich mich hier auch bei Prof. Dr. Eike Hertel dafür, dass er bereits während meiner Schulzeit mein Interesse für die Kombination von Computer und Geometrie unterstützt und gefördert und damit meinen beruflichen Weg maßgeblich beeinflusst hat.

Einleitung

Die Rekonstruktion von CAD-Modellen aus technischen Zeichnungen ist eine Aufgabenstellung, mit der sich bereits seit langer Zeit verschiedene Arbeitsgruppen weltweit beschäftigen. Wie aktuelle Veröffentlichungen belegen [1, 20], hat sie bis heute nichts an Aktualität verloren, und es stellt immer noch ein großes Problem dar, eine praktisch und produktiv einsetzbare Lösung zu entwickeln [16].

Motiviert werden Forschungsarbeiten in diesem Bereich vor allem durch zwei Gründe:

Zum einen besteht das praktische Problem, in Unternehmen vorhandene alte 2D-Zeichnungsbestände, die sowohl in Papierform als auch in Form von Dateien (etwa im DXF-Format) vorliegen können, aufzuarbeiten und die darin beschriebenen Teile für eine weitere Nutzung in aktuellen Formaten für 3D-CAD-Systeme nutzbar zu machen.

Zum anderen handelt es sich auch vom wissenschaftlichen Standpunkt aus gesehen um eine interessante, hochkomplexe Aufgabe. Sie ist ein typisches Beispiel für ein inverses Problem: Die umgekehrte Richtung, das heißt die Erstellung einer technischen 2D-Zeichnung aus einem 3D-Modell, ist im Vergleich dazu relativ leicht und wird teilweise auch als Funktionalität in aktuellen 3D-CAD-Systemen angeboten. Die Gewinnung des 3D-Modells aus einer 2D-Zeichnung erfordert jedoch das Erkennen, Verstehen und Kombinieren einer Vielzahl unterschiedlicher geometrischer, symbolischer und textueller Informationen.

Ziel dieser Arbeit ist es, im ersten Teil einen neuen Lösungsansatz für dieses Rekonstruktionsproblem vorzustellen, wobei hier ausschließlich geometrische Informationen herangezogen werden, um durch Zusammenführen der Zeichnungsinformationen in einem 3D-Voxel-Modell eine größere Robustheit gegenüber Fehlern und Qualitätsproblemen der 2D-Zeichnung zu erreichen. Gleichzeitig erhält der Nutzer die Möglichkeit, den Rekonstruktionsprozess zu beobachten und aktiv zu beeinflussen. Dieser Ansatz kann als Baustein für ein Softwaresystem zur CAD-Modell-Rekonstruktion verwendet werden.

Der zweite Teil der Arbeit beschäftigt sich ebenfalls mit Voxelmodellen, allerdings geht es hier darum, Fittingprobleme zu lösen, d. h. für eine gegebene Menge von Voxeln, die ein solides ausgefülltes Objekt repräsentieren,

eine möglichst gute Annäherung durch ein ggf. transformiertes Objekt einer vorgegebenen Objektklasse zu finden. Dazu wurde hier aufbauend auf eine Lösung für das Fitting von Rechtecken und Quadern in der Ebene ein Verfahren für Quader und Würfel entwickelt. Darüber hinaus konnte unter Nutzung einer Separation von affinen 3D-Transformationen eine Methode für das Fitting von affin transformierten Superellipsoiden gefunden werden. Beide Techniken zeichnen sich neben den Vorteilen, die die darin verwendete Normalisierungsmethode bietet, insbesondere durch eine große Robustheit gegenüber mehr oder weniger großen Störungen des Objekts wie z. B. fehlenden Ecken bei Quadern oder Löchern im Objekt aus.

Teil I

Rekonstruktion von CAD-Modellen

1 Aufgabenstellung

Ausgangsmaterial für die Rekonstruktion von 3D-CAD-Modellen sind 2D-Zeichnungen auf Papier oder in einem geeigneten Datenformat (z. B. DXF), wie sie in den Abbn. 1 und 2 zu sehen sind. Hier wird nochmals die Fülle und Verschiedenartigkeit der Informationen deutlich, die für eine vollautomatische korrekte Rekonstruktion auszuwerten sind. Anzumerken ist dabei, dass technische Zeichnungen nicht völlig frei erstellt werden, sondern eigens dafür definierten Standards folgen. Diese sind geschaffen worden, um den Datenaustausch zwischen Konstrukteuren zu normieren und damit zu erleichtern. In diesem Sinne liegen technische Zeichnungen in einer Art Code vor, der ihre Vielfältigkeit einschränkt, das „Verstehen“ dieses Codes ist aber immer noch eine hochkomplexe Aufgabe.

Das ideale Rekonstruktionsergebnis wäre ein 3D-Modell, das direkt mit einem 3D-CAD-System weiterverarbeitet werden kann. Es sollte also kein einfaches Oberflächenmodell, sondern ein aus Features[1] bestehendes Modell sein (s. a. Abschnitt 2). Abb. 3 zeigt ein Beispiel für ein solches Modell im CAD-System Autodesk Inventor. Dort sind im linken Fenster („Part Features“) verschiedene Features zu sehen, die mit diesem CAD-System konstruiert werden können. Im mittleren Fenster („Model“) sind die Struktur des Teils und die zu seiner Konstruktion verwendeten Features sichtbar.

Aus theoretischer Sicht handelt es sich bei dieser Aufgabe um eine Rekonstruktion aus orthogonalen Parallelprojektionen, bei denen zunächst keine Referenzpunkte gegeben sind. Bezüge der einzelnen Projektionen zueinander können durch geeignete Anordnung hergestellt werden. Die Ansichten sind

[1] Der Begriff Feature ist im Fachgebiet Mustererkennung verbreitet und steht dort für Merkmal. Er ist allerdings auch im CAD-Bereich fest etabliert und bezeichnet hier ein elementares Konstruktionselement bzw. eine während der Fertigung vom Materialblock zu entfernende Form.

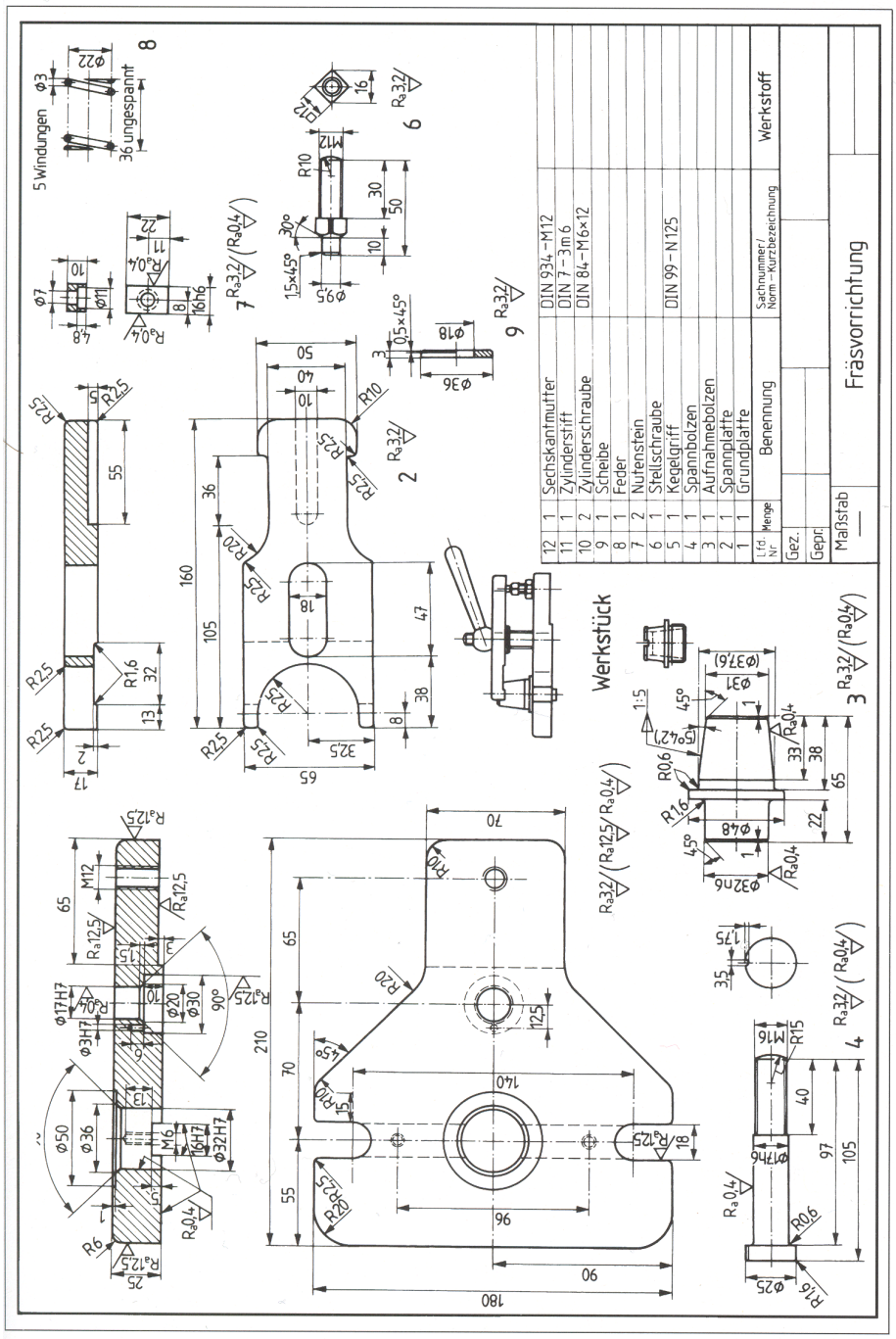


Abbildung 1: Eine komplexe CAD-Zeichnung [10].

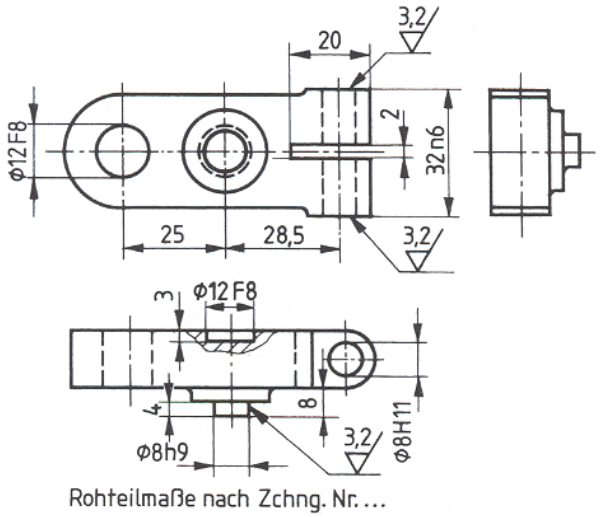


Abbildung 2: Zeichnungsausschnitt eines Einzelteils [10].

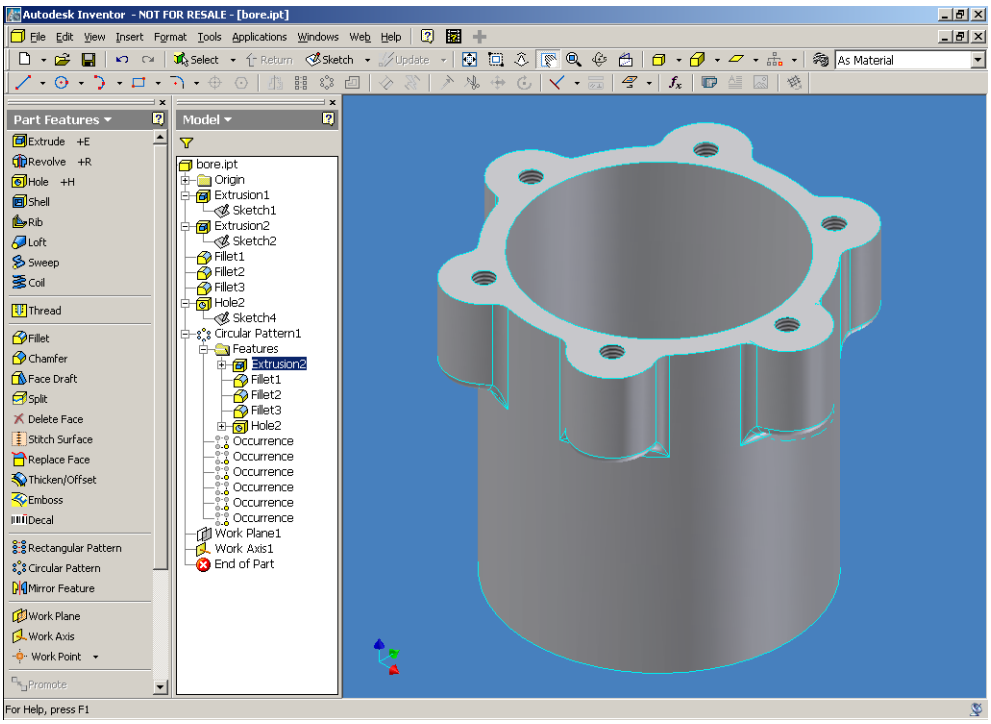


Abbildung 3: Ein 3D-Modell in Autodesk Inventor.

jedoch keine reinen Projektionen des jeweiligen Teils, sondern werden noch durch zahlreiche zusätzliche Informationen angereichert, die es vor einer Rekonstruktion zu entfernen gilt.

2 Arten von CAD-Modellen

Betrachtet man die aktuell verwendeten 3D-CAD-Modelle, so lassen sich zwei wesentliche Typen unterscheiden:

- Flächenmodelle
- Feature-orientierte Modelle

Flächenmodelle bestehen (wie der Name schon sagt) aus einzelnen Flächen, die die Oberfläche des Modells beschreiben. Daher werden sie auch oft als BRep-Modelle (Abkürzung für „boundary representation“) bezeichnet. Gebräuchlich ist hier die Verwendung von Bezier-Patches oder getrimmten Freiformflächen wie z. B. NURBS [7, 17]. Zu finden ist diese Art der Modellierung in zahlreichen Fileformaten für 3D-Daten, z. B. in DXF, IGES oder STEP, die sich als Austauschstandards etabliert haben und praktisch von allen gebräuchlichen CAD-Systemen unterstützt werden. Die Nachteile der Flächenmodelle liegen zum einen in den oft beträchtlichen Filegrößen, zum anderen auch darin, dass lediglich das Modell als Endresultat des Konstruktionsprozesses verfügbar ist. Es liegen keine Informationen vor, auf welche Weise so ein Bauteil konstruiert wurde oder zu fertigen ist.

Feature-orientierte Modelle enthalten dagegen wesentlich mehr Informationen. Die Feature-orientierte Konstruktion lässt sich kurz so charakterisieren, dass das gewünschte Modell schrittweise aus Basiselementen aufgebaut wird, die durch Parameter näher definiert werden. Solche Basiselemente können z. B. die Extrusion eines Profils entlang einer Geraden um eine bestimmte Länge oder die Rotation eines Profils um eine Achse sein (s. Abb. 4). Durch solche Elemente wird oft Material zum Modell hinzugefügt, es gibt aber auch die Möglichkeit, vom bisher konstruierten Modell Material zu entfernen. Daneben sind auch Features möglich, die eine etwas „höhere“ Funktionalität bieten, wie z. B. Features für einzelne oder mehrere Bohrungen, die in einem Gitter oder auch radial angeordnet sind. Oft ist es auch möglich, Features mittels Boolescher Operationen zu kombinieren (Vereinigung, Durchschnitt, Differenz). Ein Feature-orientiertes Modell besitzt daher im Gegensatz zu einem Flächenmodell Informationen über die Struktur des Teils (s. Abb. 3), die Datenmenge ist dabei oft deutlich geringer. Ein Nachteil dieser Art von Modellierung ist, dass solche Modelle in der Regel

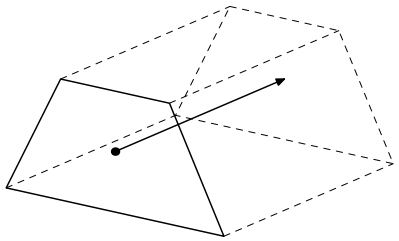


Abbildung 4: Extrusion (Austragung) eines 2D-Profils zu einem 3D-Feature

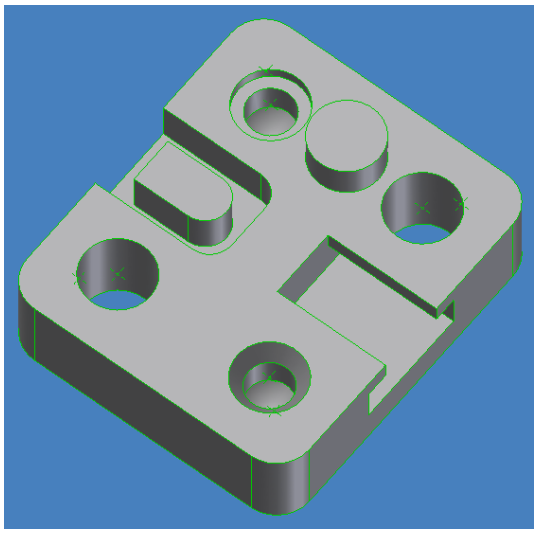


Abbildung 5: Ein Materialblock mit Beispielen für Features.

CAD-System-spezifisch sind und ein Austausch mit anderen Systemen schon aufgrund der unterschiedlichen Funktionalitäten oft unmöglich ist.

Die Bezeichnung Feature ist aber nicht nur für die Bezeichnung elementarer Konstruktionselemente gebräuchlich, sondern auch für Grundformen, die während der Fertigung von einem Materialblock zu entfernen sind, um ein Teil herzustellen („machining feature“). Der Versuch einer Standardisierung solcher Features ist z. B. im STEP-Standard[2] zu finden [30]-[34]. Abb. 5 zeigt ein Beispielteil.

[2]Standard for the Exchange of Product Model Data

3 Verfügbare Softwarelösungen

Etablierte kommerzielle Softwarelösungen zur Generierung von 3D-Modellen aus 2D-Zeichnungen sind derzeit nicht bekannt. Es gibt lediglich unterstützende Funktionen in einigen CAD-Systemen, die die Erzeugung von 3D-Modellen erleichtern. So können z. B. Konturen auf hinterlegten CAD-Zeichnungen nachgezogen und aus ihnen dann Extrusions- oder Rotationskörper generiert werden.

Derartige Funktionen gibt es z. B. in

- SolidWorks Office Professional [35],
- HiCAD [36].

Neben dem CAD-Bereich ist das Problem der 3D-Modellerzeugung auch im Architekturbereich von Bedeutung, hier gibt es ebenfalls Software, die etwa aus einem 2D-Grundriss einer Gebäudeetage durch Erkennen der Geometrie und der Symbole ein 3D-Modell erzeugt, z. B.

- Plan2Model [37],
- ACADO [38].

Darüber hinaus existieren aber zahlreiche Tools für verwandte Probleme. Zur Umwandlung von 3D-Flächenmodellen in Feature-basierte Modelle dienen z. B.

- Autodesk Feature Exchange [39],
- Solid Edge Feature Recognizer [40],
- FeatureWorks (SolidWorks) [35],
- MasterCAM Solids [41].

Zur Erkennung der Features für die Fertigung des Teils und die Erstellung des Prozessplans bzw. eines CNC-Programms gibt es z. B.

- Pro/ENGINEER Complete Machining [42],
- MasterCAM Solids [41],
- HyperMill [43],
- ProCAM II [44],
- CAMWorks [45],
- FBMach [46].

4 Bisherige Arbeiten in der Literatur

In diesem Abschnitt sollen kurz zwei in der Literatur zu findende Lösungsansätze vorgestellt werden, zum einen das Fleshing-Out-Projections-Verfahren sowie die direkte Erkennung von 3D-Features in 2D-Zeichnungen.

4.1 Das Fleshing-Out-Projections-Verfahren

Eine in der Literatur häufig zitierte Technik, die bei der Rekonstruktion aus CAD-Zeichnungen Anwendung findet, ist das sogenannte Fleshing-Out-Projections-Verfahren. Grundlage dafür sind zwei Arbeiten von Wesley und Markovsky aus den Jahren 1980 und 1981. Die Autoren beschäftigen sich in [26] mit dem Problem, zu einem vorgegebenen Drahtmodell ein entsprechendes Volumenmodell zu berechnen. Zur damaligen Zeit waren Volumenmodelle in der Konstruktion längst nicht so gebräuchlich wie heute, wurden aber gelegentlich benötigt, um verschiedene Berechnungen (Festigkeit, Wärmeleitung, Erstellung von NC-Programmen für die Fertigung) durchzuführen. In ihrer zweiten Arbeit [27] werden als Eingangsdaten nicht das Drahtmodell selbst, sondern zwei oder mehr Projektionen davon verwendet. Aus ihnen wird zunächst eine Vorstufe des eigentlichen Drahtmodells berechnet, auf die dann eine modifizierte Variante des Verfahrens aus [26] angewendet wird, um das Volumenmodell zu bilden. Damit ist diese zweite Arbeit auch für die Rekonstruktion von Teilen aus CAD-Zeichnungen interessant.

Der in [26] angegebene Algorithmus konstruiert zu einem aus Ecken und Kanten bestehenden Drahtmodell alle diejenigen Polyeder, deren Drahtmodell mit dem vorgegebenen übereinstimmt. Die Möglichkeit, dass mehrere Lösungen existieren, ist dabei ausdrücklich eingeschlossen und durch Beispiele belegt. Der Algorithmus ist in der Lage, unter Verwendung einer exhaustiven Suche alle existierenden Lösungen zu generieren. Das Verfahren läuft dabei in folgenden wesentlichen Schritten ab (s. Abb. 6):

- Finden möglicher Trägerebenen für Seitenflächen,
- Generierung möglicher Seitenflächen, sog. virtueller Flächen,
- Finden von Schnitten zwischen virtuellen Flächen und ggf. weitere Aufteilung der virtuellen Flächen,

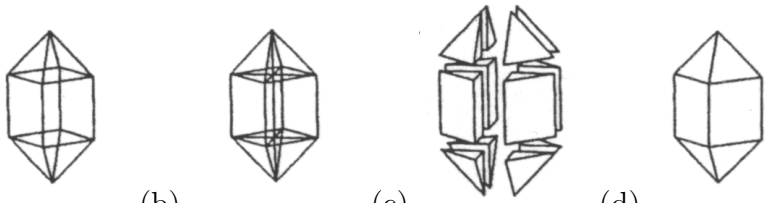


Abbildung 6: (a) Drahtmodell, (b) Drahtmodell mit neuen Kanten, die durch Aufteilung sich schneidender virtueller Flächen entstanden sind, (c) entstehende virtuelle Blöcke, (d) Lösung [26].

- Generierung von kleinsten Basiselementen (virtuellen Blöcken), die durch die virtuellen Flächen begrenzt werden,
- Erzeugen aller möglichen Lösungen durch Zuweisung der Zustände leer/gefüllt an die virtuellen Blöcke.

In [27] wird eine leicht modifizierte Version dieses Algorithmus zur Rekonstruktion von Polyedern, von denen zwei oder mehr senkrechte Parallelprojektionen (paarweise verschiedene Richtungen) gegeben sind, benutzt. Dabei sind alle Kanten unabhängig von ihrer Sichtbarkeit zu projizieren. Dem oben beschriebenen Prozess werden noch weitere Schritte vorangestellt, um aus den Projektionen ein Kandidaten-Drahtmodell zu erzeugen. Diese Bezeichnung rührt daher, dass in ihm auch Elemente enthalten sind, die nicht Bestandteile des Drahtmodells für das bzw. die Lösungspolyeder sind. Durch die Verwendung der Projektionen werden erneut Mehrdeutigkeiten eingebracht, die zu einer weiteren Vergrößerung des Suchaufwands führen. Es sind einige Beispiele angegeben, die eine große Vielfalt von Lösungen zulassen (Abb. 7), ebenso aber auch Beispiele von im ingenieurtechnischen Bereich vorkommenden Körpern mit einer eindeutigen Lösung (Abb. 8). Die Mehrdeutigkeit des Beispiels in Abb. 7 könnte sicherlich noch durch eine zusätzliche Ansicht eingeschränkt werden, aber es gibt auch Beispiele, bei denen auch drei Ansichten keine eindeutige Lösung liefern. Nach Angabe der Autoren ist die Laufzeit des Algorithmus proportional zur Anzahl der Lösungen, und für „reale“ Objekte werden die Lösungen in vertretbarer Zeit gefunden, da die Geometrie des Objekts im Gegensatz zu theoretisch konstruierten Beispielen mit einer großen Anzahl von Lösungen den Suchaufwand sehr schnell einschränkt.

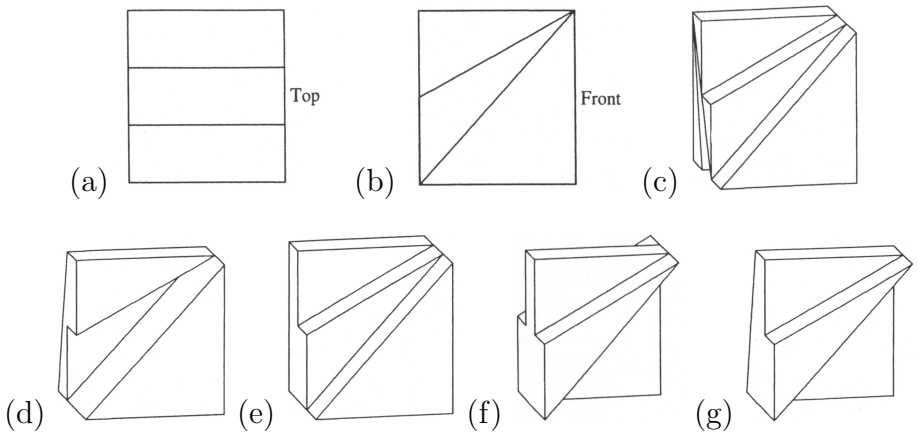


Abbildung 7: Zwei Projektionen und fünf der daraus erzeugbaren 107 Lösungen [27].

Die beiden angegebenen Algorithmen arbeiten ausschließlich mit geradlinig begrenzten Objekten, eine Ausweitung dieses Prinzips auf andere Arten von Begrenzungsflächen ist aber durchaus möglich und wird in weiteren Arbeiten auch vorgenommen, so z. B. in [14]. Der dort angegebene Algorithmus ist in der Lage, durch Quadriken begrenzte Körper zu rekonstruieren. Die Autoren geben einige erfolgreich rekonstruierte Beispiele an, die benötigte Rechenzeit lag zur Zeit der Veröffentlichung im Bereich von wenigen Sekunden.

Daneben existieren noch zahlreiche Arbeiten, die sich mit der Optimierung des Suchaufwandes beschäftigen, z. B. [29].

Der Vorteil dieser Methode ist das Finden aller möglichen Lösungen, Bedingung ist dafür aber eine perfekte Qualität der Ausgangsdaten, was die direkte Anwendung auf Papierzeichnungen bzw. deren Vektorisierungsergebnisse erschwert.

Einen weiteren Nachteil stellt der Suchprozess dar. Zum einen ist schwer abschätzbar, wieviel Zeit er bei der Rekonstruktion von komplexen Modellen wirklich in Anspruch nimmt (in [15] findet sich die Aussage, dass diese Methode für Objekte aus dem „wirklichen Leben“ nicht einsetzbar ist). Zum anderen ist dieser Suchprozess für einen Benutzer ohne Detailkenntnisse über die Arbeitsweise des Verfahrens sicherlich unverständlich. Es dürfte also schwerfallen, dem Nutzer z. B. eine Bedienoberfläche anzubieten, über die

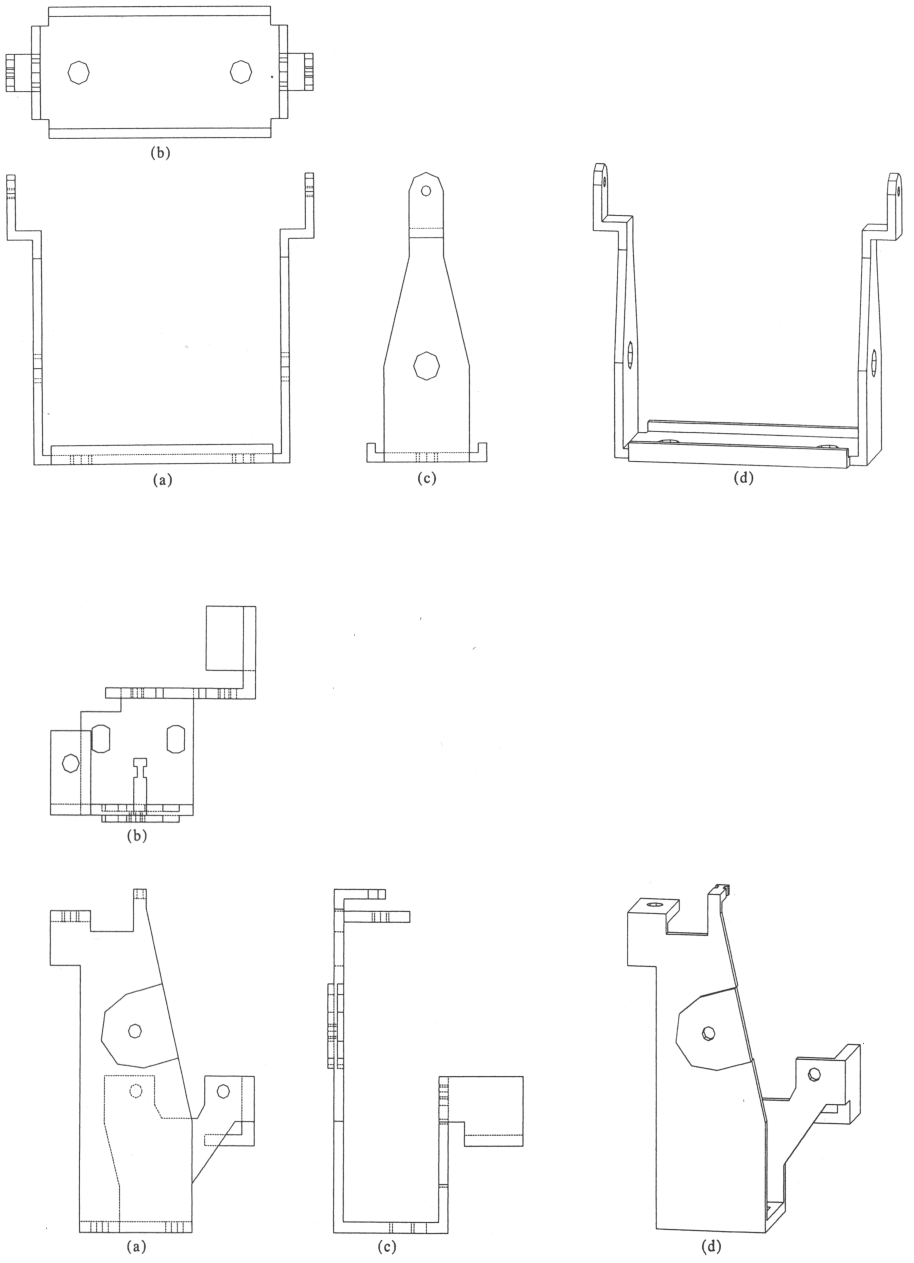


Abbildung 8: Zwei erfolgreich rekonstruierte technische Objekte [27].

er interaktiv die Erkennung beeinflussen kann. Letzteres sollte aber möglich sein, da in der praktischen Anwendung immer mit Problemen und Fehlern zu rechnen ist, die einer manuellen Korrektur bedürfen.

Abgesehen davon liefert das Verfahren lediglich ein Flächenmodell, das zur Weiterverarbeitung im CAD-System nur bedingt geeignet ist.

4.2 2D-Feature-Erkennung

Zahlreiche Arbeiten beschäftigen sich damit, direkt aus den 2D-Ansichten die Features zu extrahieren, aus denen ein Modell aufgebaut ist bzw. die Features zu finden, die für die Fertigung relevant sind. Die darin vorgeschlagenen Lösungen setzen bestimmte Eigenschaften der Objekte voraus und sind deswegen nicht allgemein einsetzbar. Im Gegensatz zum Fleshing-Out-Projections-Verfahren, hinter dem ein Algorithmus zur Generierung aller möglichen Lösungen steht, basieren diese Verfahren auf Heuristiken und liefern keine vollständige Aufzählung der Lösungen.

Um eine Vorstellung von der Arbeitsweise zu vermitteln, werden in diesem Abschnitt drei ausgewählte Verfahren kurz vorgestellt und an jeweils einem Beispiel erläutert.

In [18] wird ein zweistufiges Verfahren zur Rekonstruktion aus drei Ansichten beschrieben: Zunächst werden aus den Außenkonturen und eventuell vorhandenen Innenkonturen der drei Ansichten drei Extrusionskörper gebildet, diese werden zum Schnitt gebracht und bilden einen Basiskörper. Die weiteren „inneren“ Linien der Ansichten werden daraufhin untersucht, ob sich aus ihnen noch weitere Extrusionskörper bilden lassen, die dann gegebenenfalls vom Basiskörper abgezogen werden (Abb. 9 und 10). Die Funktionsfähigkeit der Methode ist auf Objekte beschränkt, die aus Extrusionen erzeugt werden können. Ein Beispiel, wo sie nicht eingesetzt werden kann zeigt die Abb. 12, dies liegt hier aber auch an der Mehrdeutigkeit der dargestellten Projektionen und der Tatsache, dass hier keine „echten“ Körperkanten projiziert wurden.

Meeran und Taib beschreiben in [15] ein System, das ebenfalls Extrusionsfeatures finden kann. Die Ansichten des Objekts werden durch Graphen repräsentiert. In ihnen wird nach drei vordefinierten Merkmalen gesucht, die auf die Existenz von Features hinweisen. Anschließend werden die gefundenen

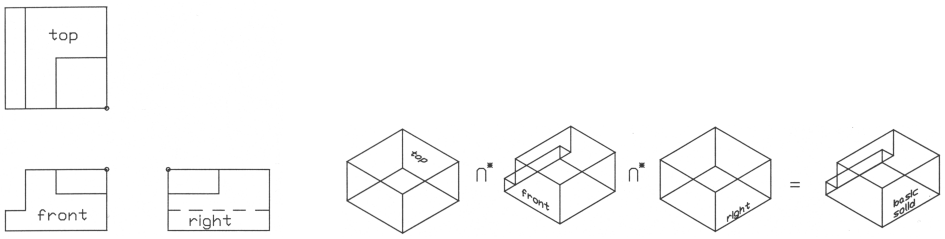


Abbildung 9: Drei Ansichten eines Körpers und der daraus erzeugte Basiskörper [18].

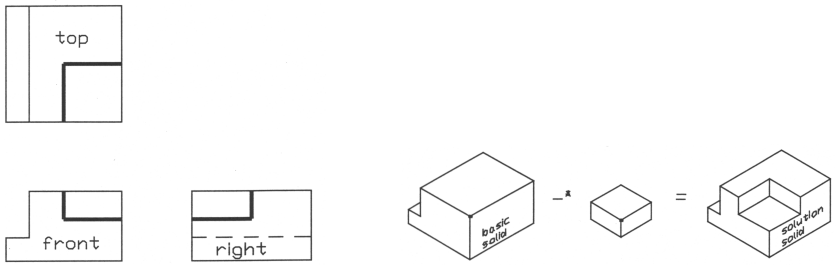


Abbildung 10: Entfernen eines weiteren Extrusionskörpers liefert den Ergebniskörper [18].

Merkmale durch Hinzufügen der Tiefeninformation zu Features komplettiert. Abb. 13 zeigt einige komplexe Beispiele, bei denen das Verfahren erfolgreich ist. Allerdings ist es auf achsenparallele Features beschränkt.

Ein weiteres System zur Featureerkennung wird in [8] vorgestellt. Hier werden aus den drei Ansichten sukzessive Features extrahiert, zunächst beginnend mit isolierten Features wie einzelnen Durchgangsbohrungen. Daran anschließend werden Features verarbeitet, die sich mit anderen Features schneiden oder gemeinsame Kanten mit der Außenkontur des Teils haben, z. B. Verrundungen, Fasen oder abgesetzte Ecken. In Abb. 14 wird dies schrittweise an einem Beispiel demonstriert.

Allen diesen Lösungen ist gemeinsam, dass sie ebenso wie das Fleshing-Out-Projections-Verfahren perfekte Eingangsdaten voraussetzen, also für die Anwendung auf Papierzeichnungen bzw. deren Vektorisierungsergebnissen nur bedingt geeignet sind. Charakteristisch ist weiterhin, dass sie nur für spezielle Teile bzw. Features einsetzbar sind, und auch dann ist nicht sichergestellt, dass jedes Teil erfolgreich rekonstruierbar ist.

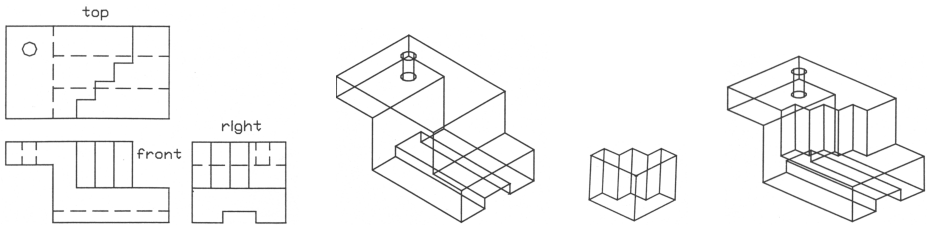


Abbildung 11: Drei Ansichten eines weiteren Teils, der daraus entstehende Basiskörper, ein zu entfernender Extrusionskörper und das Endergebnis [18].

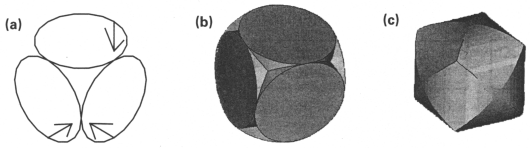


Abbildung 12: Eine Kugel liefert in jeder Projektion einen Kreis, der Durchschnitt dreier Zylinder ist aber keine Kugel [18].

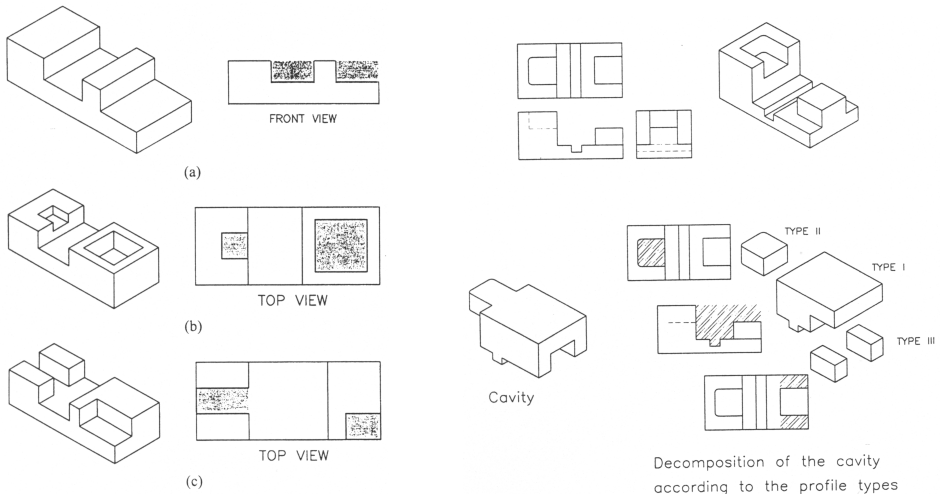


Abbildung 13: Links: Die drei verschiedenen Profiltypen. Rechts: Ein Teil mit 3 Projektionen, der zu entfernende Materialblock und seine Zusammensetzung aus den drei Profiltypen [15].

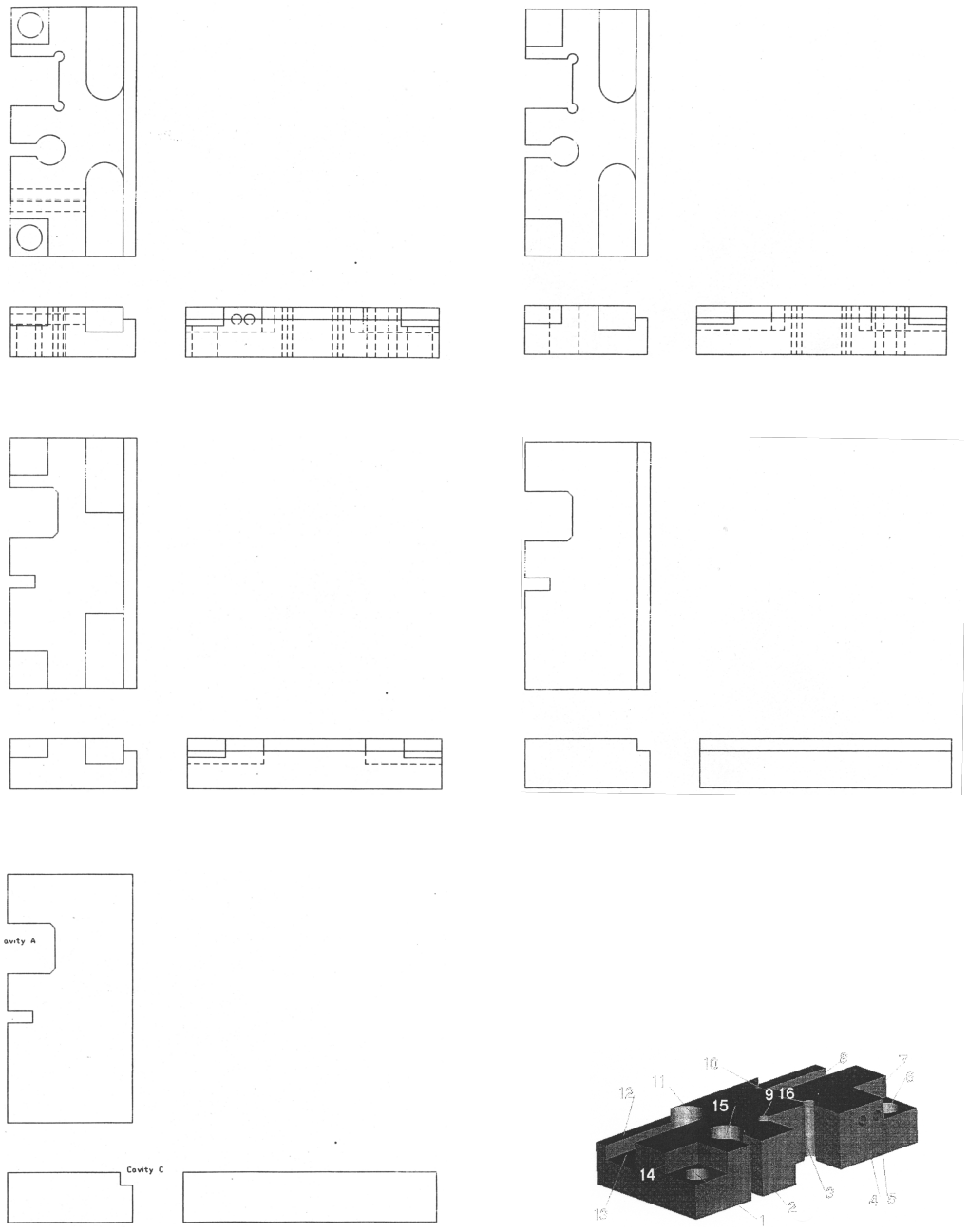


Abbildung 14: Schrittweise Erkennung und Extraktion von Features aus den drei Ansichten sowie das resultierende 3D-Modell [8].

5 Voxelbasierte Rekonstruktion

Die in den Abschnitten 4.1 and 4.2 vorgestellten Verfahren setzen für eine erfolgreiche Rekonstruktion eine perfekte Qualität der Eingangsdaten voraus, d. h. es wird gefordert, dass die Daten so genau sind, als ob sie direkt aus einem existierenden, korrekten 3D-Modell entstanden sind.

Eine solche Forderung ist allerdings in der Praxis kaum zu erfüllen. Bei Zeichnungen, die mit einem 2D-CAD-System erzeugt wurden, mag es unter Umständen noch möglich sein, eine solche Qualität annähernd zu erreichen. Allerdings zeigt die Erfahrung, dass auch hier immer mit Ungenauigkeiten zu rechnen ist (s. a. [16]).

Werden Papierzeichnungen verarbeitet, ist die perfekte Datenqualität erst recht illusorisch. Bisher war es üblich, solche Zeichnungen zunächst zu vektorisieren, um dann mit Vektordaten weiterarbeiten zu können. Neben den zahlreichen Problemen, die die Vektorisierung selbst mit sich bringt (mangelnde Qualität, Unexaktheiten, unterbrochene Linien, die eigentlich eine einzige Linie bilden etc.), ist es natürlich auch so, dass die geforderte Genauigkeit auch in der Zeichnung selbst schlichtweg nicht vorhanden ist. So kann etwa das genaue Maß für die Dicke eines Objekts der Skizze selbst nicht als Abstand zweier Linien (die nur als Menge von Pixeln unterschiedlicher Intensität vorliegen) entnommen werden, sondern dazu ist die inhaltliche Auswertung der textuellen Information (hier der Bemaßung) notwendig.

In diesem Abschnitt wird eine Lösung präsentiert, die als Komponente eines Erkennungssystems eingesetzt werden kann, und mit der versucht wird, den oben angesprochenen Problemen teilweise zu begegnen. Dabei werden zunächst nur die Geometriedaten betrachtet, die zu den Projektionen der Teilegeometrie gehören. Auf ein Hinzuziehen und Auswerten weiterer Informationen wie Bemaßungen oder Texte wird an dieser Stelle verzichtet. Grundlage dafür ist die neue, in der Literatur bisher nicht vorzufindende Idee, direkt aus den gescannten Zeichnungen ohne Verwendung einer Vektorisierung ein 3D-Modell zu erstellen, welches die Struktur des zu rekonstruierenden Teils gut wiedergibt (s. [2, 3]). Die Software ist als Unterstützung für den Bearbeiter gedacht, dessen Aufgabe die Rekonstruktion eines 3D-Modells aus 2D-Ansichten ist. Sie bietet keine vollautomatische Rekonstruktion, sondern der Nutzer wird in den Rekonstruktionsprozess aktiv mit einbezogen.

5.1 Voxelmodell

Die Grundlage für die Lösung bildet eine Eigenschaft des während des Fleshing-Out-Projections-Verfahren entstandenen Kandidaten-Drahtmodells. Die Abb. 15 und 16 zeigen eine Vorstufe dieser Modelle für eine durch Facetten angenäherte Kugel und ein fiktives CAD-Teil. Entstanden sind diese Modelle auf folgende Weise:

- Wir nehmen an, die Projektionen liegen auf den im Raum wie bei ihrer Entstehung angeordneten Projektionsebenen.
- In einem ersten Schritt werden über allen Eckpunkten der Projektionen Senkrechten errichtet. Schneiden sich zwei Senkrechten aus verschiedenen Projektionen in einem Punkt, so wird geprüft, ob die Projektion dieses Punktes mit den übrigen Projektionen verträglich ist, d. h. ob die Projektion dieses Punktes auf die anderen Ebenen mit einer Ecke zusammenfällt oder auf einer Kante liegt. Ist das der Fall, wird diese Ecke als Kandidat in das Drahtmodell aufgenommen.
- In einem zweiten Schritt wird jeweils zwischen zwei im ersten Schritt erzeugten Ecken eine Kante in das Drahtmodell aufgenommen, wenn diese Kante mit allen Projektionen verträglich ist, d. h. dort zu einem Punkt zusammenfällt, der mit einer Ecke übereinstimmt oder auf einer Kante der Projektion liegt, oder eine Kante bildet, die wiederum Teil einer Projektionskante ist oder mit ihr übereinstimmt.

Zur vollständigen Berechnung des Kandidaten-Drahtmodells werden dann noch weitere Schritte ausgeführt, etwa das Entfernen überflüssiger Elemente, wie z. B. von Ecken, die nur mit zwei Kanten inzident sind und somit nicht zum Modell gehören können. Darauf wurde hier jedoch verzichtet.

Auffallend ist, dass für den menschlichen Betrachter das Kandidaten-Drahtmodell bereits sehr große Ähnlichkeit mit dem tatsächlichen Teil hat und als Teil einer Softwarelösung genutzt werden kann, um dem Benutzer einen ersten Eindruck von diesem Teil zu bieten.

Es ist denkbar, bereits auf diese Daten Algorithmen zur 3D-Featureerkennung anzuwenden, um die Teilgeometrie zu rekonstruieren.

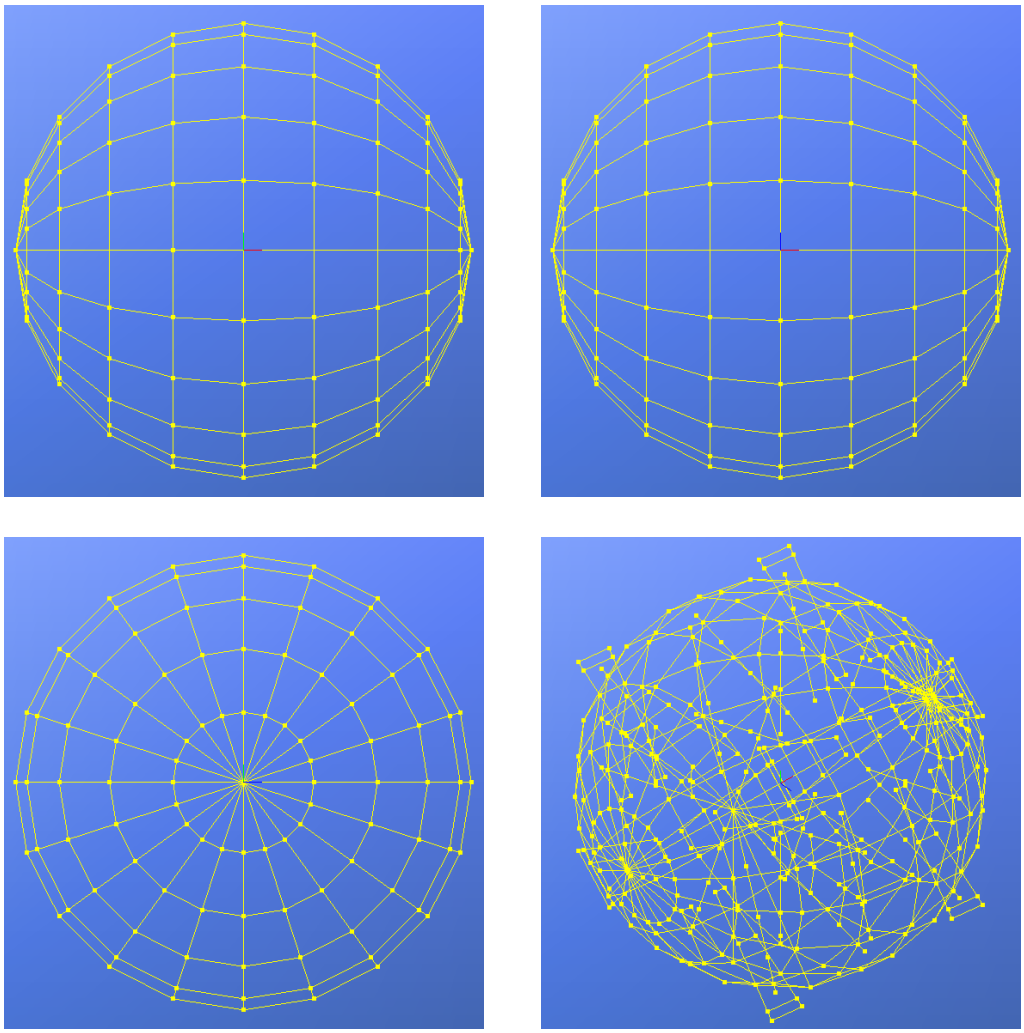


Abbildung 15: Drei Projektionen einer durch Facetten angenäherten Kugel und das daraus erzeugte Kandidaten-Drahtmodell.

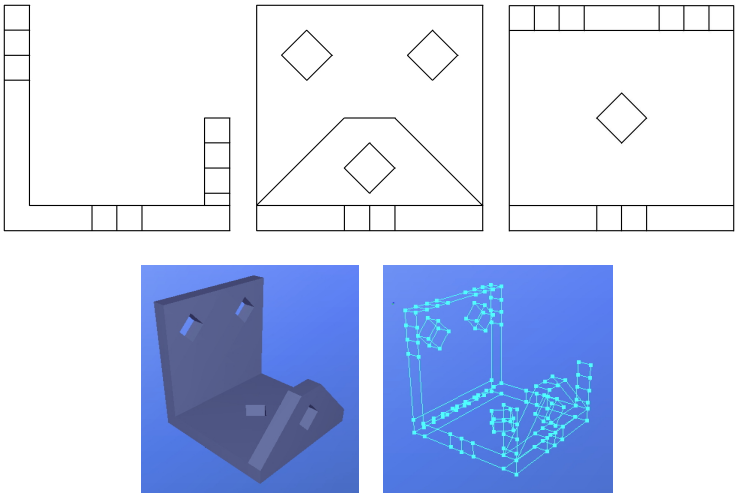


Abbildung 16: Ein CAD-Teil (links unten), drei Projektionen (oben) und das daraus erzeugte Kandidaten-Drahtmodell (rechts unten).

Für dieses Kandidaten-Drahtmodell sind natürlich ebenfalls Vektordaten, also Ecken und Kanten als Eingaben nötig. Um auch Papierzeichnungen bearbeiten zu können, kann man versuchen, unter Verzicht auf eine vorherige Vektorisierung die Informationen aus den gescannten Bitmaps direkt im 3D-Raum zusammenzuführen. Diese Idee bildet das Kernstück der hier vorgestellten Lösung und schafft die Verbindung zwischen den im Fleshing-Out-Projections-Algorithmus verwendeten Techniken auf der einen und den weiter unten beschriebenen Methoden zur Akkumulation bzw. zum Finden von Features und der darauf aufbauenden Kombination der Features zu Teilen auf der anderen Seite. Im folgenden wird die Vorgehensweise zur Bildung des Voxelmodells beschrieben.

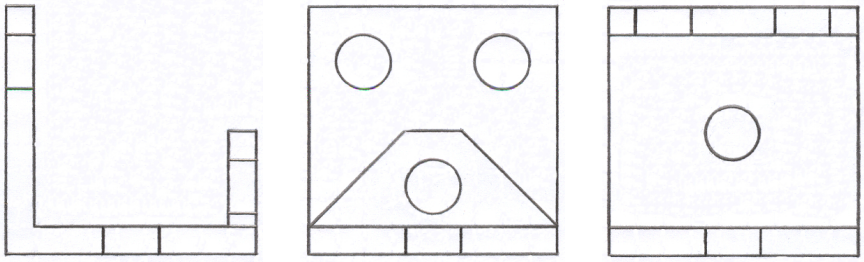


Abbildung 17: Drei gescannte Ansichten eines CAD-Teils.

Abb. 17 zeigt drei von Papier gescannte Ansichten des (leicht modifizierten) CAD-Teils aus Abb. 16. Diese Projektionen werden nun auf die XY-, XZ- und YZ-Ebenen eines 3D-Koordinatensystems gelegt. Dies soll so geschehen, dass sie korrekt zueinander ausgerichtet sind, so dass z. B. die X-Koordinaten der XY- und XZ-Projektionen eines 3D-Punktes übereinstimmen (analog für die Y- bzw. Z-Koordinaten). Aus diesen Projektionen wird nun durch Addition und anschließende Normierung der Intensitäten eine 3D-Voxelstruktur erzeugt (Abb. 18).

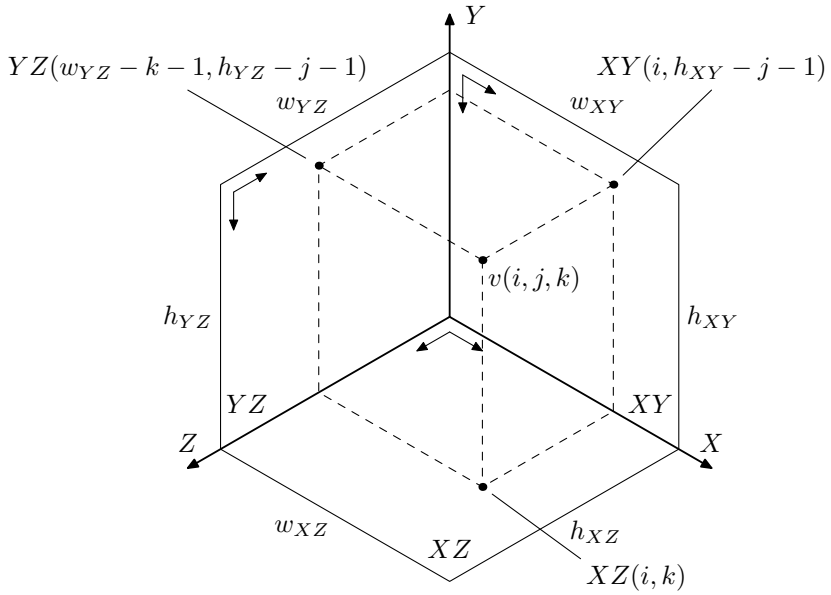


Abbildung 18: Berechnung der Voxelintensitäten.

Für ein Bild B der Größe $w_B \times h_B$ sei

$$B(i,j) \quad \text{mit} \quad i \in [0, w_B - 1],\ j \in [0, h_B - 1]$$

die Intensität des Pixels mit der Position (i, j). Für die Bilder

$$XY,\ YZ,\ XZ \quad \text{mit} \quad w_{XY} = w_{XZ},\ h_{XY} = h_{YZ},\ w_{YZ} = h_{XZ}$$

wird die Intensität des Voxels $V(i, j, k)$ wie folgt ermittelt:

Es werden die Werte

$$v(i,j,k) = XY(i, h_{XY} - j - 1) + XZ(i,j) + YZ(w_{YZ} - k - 1, h_{YZ} - j - 1)$$

für alle

$$i \in [0, w_{XY} - 1],\ j \in [0, h_{XY} - 1],\ k \in [0, w_{YZ} - 1]$$

berechnet und anschließend auf den Bereich $[0, 1]$ normiert, d. h. wir erhalten

$$V(i,j,k) = \begin{cases} \dfrac{v(i,j,k)}{v_{max}} & \text{falls } v_{max} \neq 0 \\ 0 & \text{falls } v_{max} = 0 \end{cases} \qquad \text{mit } v_{max} = \max_{\substack{i \in [0, w_{XY}-1] \\ j \in [0, h_{XY}-1] \\ k \in [0, w_{YZ}-1]}} v(i,j,k)$$

Die Abb. 19 zeigt das Histogramm der entstehenden Voxel-Intensitäten für die Projektionen aus Abb. 17 (mit Faktor 50 in y-Richtung skaliert). Deutlich sind vier Bereiche zu unterscheiden: 0.0 bis 0.5, 0.5 bis 0.7, 0.7 bis 0.9 und 0.9 bis 1.0. Sie können so interpretiert werden, dass sie verschiedene Teile des Voxelmodells enthalten. Angenommen, in den Projektionen gäbe es nur die Intensitäten 0 und 1. Dann würde die resultierende Voxelstruktur nur die Intensitäten 0, $\frac{1}{3}$, $\frac{2}{3}$ und 1 enthalten, und diese Zahlen würden beschreiben, wieviele Pixel der Intensität 1 sich in einem Voxel „treffen“. Voxel mit der geringsten Intensität treten dort auf, wo sich drei Pixel geringer Intensität (d. h. drei Punkte der Projektionen) treffen und Voxel mit hoher Intensität erscheinen dort, wo drei Pixel mit hoher Intensität zusammentreffen (d. h. Hintergrundpixel aus den Projektionen). Wählen wir für unser Beispiel den Intensitätsbereich 0 bis 0.45 aus, erhalten wir die in Abb. 20 abgebildete Struktur.

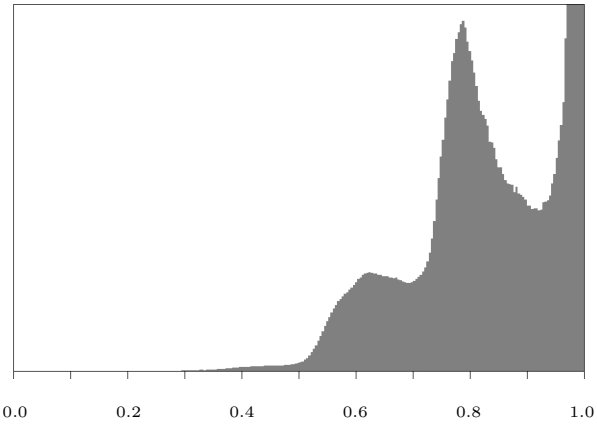


Abbildung 19: Histogramm der entstehenden Voxelintensitäten (in y-Richtung mit Faktor 50 skaliert).

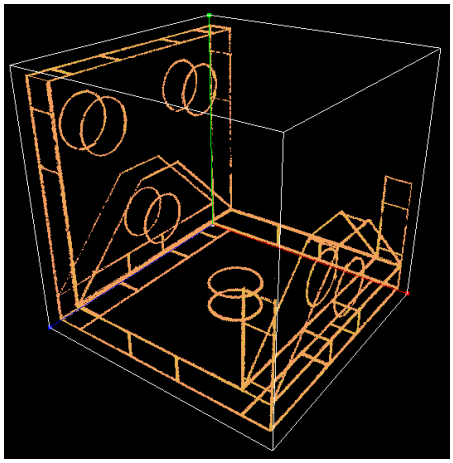


Abbildung 20: Voxel mit einer Intensität von 0.0 bis 0.45 für die Projektionen aus Abb. 17.

Ganz offensichtlich besteht eine große Ähnlichkeit zwischen dem oben gezeigten Kandidaten-Drahtmodell und der hier entstandenen Voxelstruktur. Tatsächlich lässt sich für ausschließlich durch gerade Kanten begrenzte Objekte zeigen, dass die Voxelstruktur das Kandidaten-Drahtmodell in gewisser Weise enthält (Abb. 21). Angenommen, wir generieren zwei Voxelstrukturen M_1 und M_2. M_1 wird erzeugt, indem das Kandidaten-Drahtmodell berechnet und anschließend in das Voxelraster digitalisiert wird, indem jedes Voxel, das von einer Kante geschnitten wird, die Intensität 0 erhält und alle übrigen die Intensität 1. M_2 entsteht, indem zuerst die Bilder digitalisiert werden, und anschließend erfolgt die Addition der Intensitäten wie oben beschrieben. Hat nun ein Voxel in M_1 den Wert 0, so muss dieses Voxel von einer Kante geschnitten worden sein. Das bedeutet aber, dass diese Kante mit den Projektionen verträglich war und somit auch die entsprechenden Pixel in den digitalisierten Projektionen den Wert 0 haben müssen. Damit hat dieses Voxel aber auch in M_2 die Intensität 0. Demzufolge sind also alle Informationen aus dem Kandidaten-Drahtmodell, genauer gesagt alle Kanten des Objekts, auch in der durch die Addition konstruierten Voxelstruktur enthalten. Das rechtfertigt den Einsatz dieser Struktur als Basis für die 3D-Rekonstruktion eines CAD-Modells.

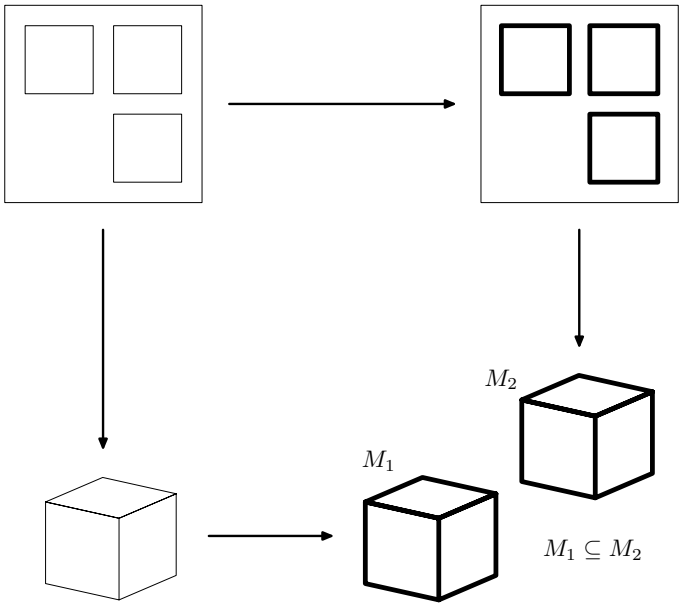


Abbildung 21: Voxelstruktur enthält Kandidaten-Drahtmodell.

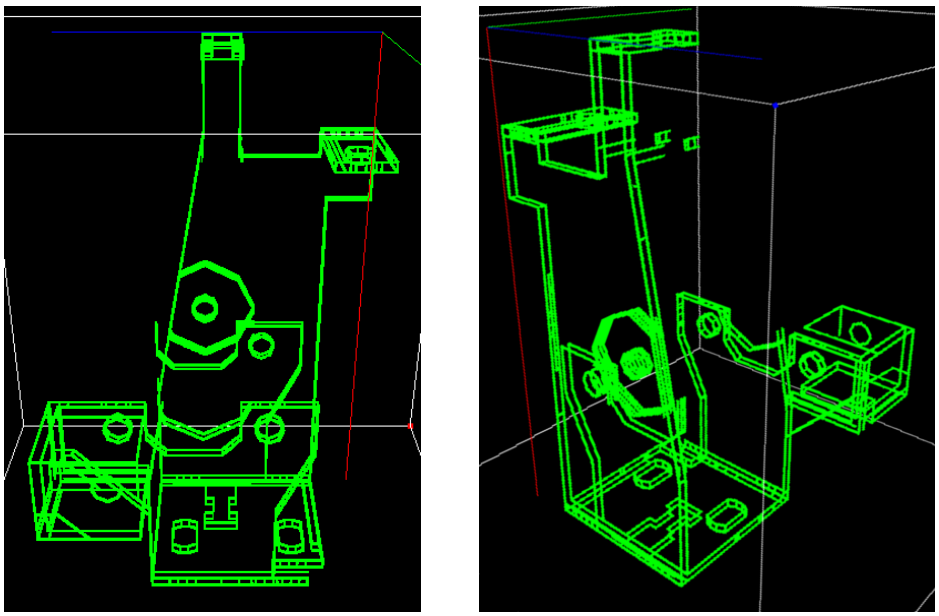


Abbildung 22: Voxelstruktur für ein Teil aus Abb. 8

5.2 Finden von Features mittels Akkumulation

Die oben konstruierte Voxelstruktur kann nun in weiteren Schritten genutzt werden, um aus ihr ein zu den vorgegebenen Projektionen passendes 3D-CAD-Modell zu rekonstruieren. Wie schon in Abschnitt 1 erwähnt, sollte das Endergebnis kein Flächenmodell, sondern möglichst ein featurebasiertes Modell sein, das in einem CAD-System direkt weiterverwendet werden kann.

Daher wird nun versucht, in der entstandenen Struktur solche CAD-Features zu finden. Als eine Möglichkeit dafür verwenden wir eine neu entwickelte Akkumulationstechnik (veröffentlicht in [2]), die auf der randomisierten Hough-Transformation (RHT, [28]) basiert.

5.2.1 Akkumulation

Das Grundprinzip soll hier kurz erläutert werden: Die (konventionelle) Hough-Transformation [12] kann z. B. zum Finden von Linien in 2D-Bildern benutzt werden. Ist eine Menge von Punkten gegeben, so werden für jeden Punkt $P = P(x, y)$ diskrete Punkte (p, φ) der Kurve $p = x \cos\varphi + y \sin\varphi$ in einem Parameterraum akkumuliert, d. h. wir zerlegen den Parameterraum in eine diskrete Menge von Zellen (das Hough-Bild) und zählen für jede Zelle die Anzahl der Treffer. Anschließend wird in diesem Hough-Bild nach Peaks gesucht (Abb. 23). Die Hough-Transformation akkumuliert für jeden Punkt P alle Parametertupel für Linien, die durch diesen Punkt verlaufen, in [28] wird dies als „diverging mapping“ bezeichnet, da für einen Punkt mehrere Parametertupel berechnet und akkumuliert werden. Im Gegensatz dazu verwendet die RHT mehrere Punkte und berechnet und akkumuliert ein einzelnes Parametertupel („converging mapping“). Für Linien würden jeweils zwei Punkte gewählt werden, um aus ihnen die beiden Parameter zu bestimmen. Mit dieser Methode werden die Peaks im RHT-Bild deutlich schärfer, da in diesem Fall das RHT-Bild die quadrierten Werte des Hough-Bildes enthält. In [28] wird weiterhin gezeigt, dass es nicht nötig ist, über alle Punktepaare zu akkumulieren, sondern es genügt, eine geeignete Anzahl zufällig gewählter Punktepaare zu verwenden, wodurch sich die Laufzeit deutlich verringert.

Die RHT kann auch für andere Objekte benutzt werden, die sich durch Parametertupel beschreiben lassen. Der Nachteil dabei ist, dass mit zunehmender Parameteranzahl auch größere Akkumulationsräume benötigt werden, in denen aber nur sehr kleine Regionen wirklich verwendet werden. Eine

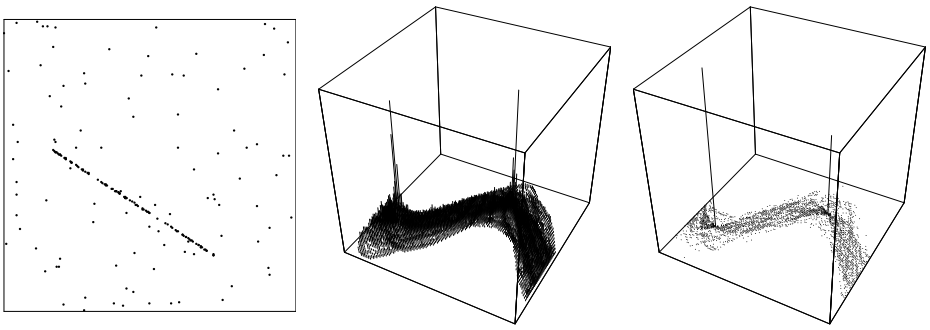


Abbildung 23: Eine Linie mit einigen zusätzlichen Punkten und das resultierende Hough- und RHT-Bild (das Hough-Bild ist gegenüber dem RHT-Bild um den Faktor 25 vergrößert).

Lösungsmöglichkeit für dieses Problem besteht darin, die Akkumulation nicht im Akkumulationsraum selbst, sondern in niederdimensionalen Projektionen auszuführen.

Diese Methode hat wiederum den Nachteil, dass durch die Projektion natürlich Information verlorengeht. Durch das Aufaddieren der Parametertupel entlang der Projektionsrichtungen kann es sehr schwer oder unmöglich sein, überhaupt Peaks zu finden. Außerdem ist es möglich, dass dadurch Pseudo-Peaks entstehen.

Eine weitere Lösungsmöglichkeit besteht in der Anwendung eines Clusterings: Anstatt einen Akkumulator zu verwenden, wird eine Liste von Clustern verwaltet. Diese Methode wird hier zum Finden von CAD-Elementen in der Voxelstruktur verwendet.

Die Eingabe besteht aus einer Menge von 3D-Punkten, die aus dem Voxelmodell durch Auswahl eines Intensitätsbereichs erzeugt wird. Ergebnis des Verfahrens ist eine Liste akkumulierter Parametertupel, jedes der Tupel ist dabei mit einem Gewicht und einem Qualitätsmaß versehen. Der Algorithmus wird durch drei Funktionen und drei Werte gesteuert:

- Eine Testfunktion $t(\mathbf{p})$ mit den Rückgabewerten „valid“ und „invalid“, die überprüft, ob ein Parametertupel $\mathbf{p}$ ein gültiges Objekt beschreibt,

- eine Qualitätsfunktion $\mu(\mathbf{p})$, die ein geeignet definiertes Qualitätsmaß für ein durch $\mathbf{p}$ beschriebenes Objekt liefert,
- eine Distanzfunktion $d(\mathbf{p}, \mathbf{q})$, die den Abstand zweier Parametertupel zurückgibt,
- die maximale Anzahl $step_{max}$ auszuwählender Punktetupel,
- die minimale Qualität μ_{min} für Objekte, die akkumuliert werden sollen,
- den minimalen Abstand d_{min}, für den zwei Tupel als verschieden betrachtet werden.

Der Ablauf des Verfahrens ist folgender:

1. Setze $step = 0$.
2. Erhöhe $step$ um 1. Falls $step > step_{max}$, ist der Algorithmus beendet.
3. Wähle zufällig n Punkte und berechne das zugehörige Parametertupel $\mathbf{p}$.
4. Falls $t(\mathbf{p})$ den Wert „invalid“ liefert, gehe zu 2.
5. Falls $\mu(\mathbf{p})$ einen Wert kleiner als μ_{min} liefert, gehe zu 2.
6. Falls die Liste der akkumulierten Cluster leer ist, initialisiere sie mit $\mathbf{p}$ als erstem Element mit Gewicht 1 und gehe zu 2.
7. Suche in der Akkumulationsliste ein Tupel $\mathbf{q}$ mit $d(\mathbf{p}, \mathbf{q}) < d_{min}$. Falls es mehrere davon gibt, wähle eines mit dem geringsten Abstand.
8. Falls ein Tupel $\mathbf{q}$ gefunden wurde (mit dem Gewicht $w_{\mathbf{q}}$), ersetze es durch $\mathbf{q}' = (w_{\mathbf{q}}\mathbf{q} + \mathbf{p})/(w_{\mathbf{q}} + 1)$ mit dem neuen Gewicht $w_{\mathbf{q}'} = w_{\mathbf{q}} + 1$ und gehe zu 2.
9. Falls kein Tupel gefunden wurde, füge $\mathbf{p}$ mit dem Gewicht 1 als neues Tupel zur Akkumulationsliste hinzu und gehe zu 2.

Nicht nur elementare Geometrieelemente wie Linien oder Ebenen, sondern insbesondere auch andere komplexe Objekte, für die keine Beschreibung in analytischer Form möglich ist, können mit dieser Technik gefunden werden. Damit ist sie geeignet, in der vorliegenden Voxelstruktur CAD-Features zu identifizieren. In den folgenden beiden Abschnitten wird gezeigt, wie mit der beschriebenen Technik als Grundlage auch diese 3D-Formen gefunden werden können, hier werden Quader und Zylinder mittels Akkumulation in der Voxelstruktur erkannt. Von Bedeutung sind dabei besonders eine geeignete Definition des Parametertupels, welches das jeweilige Objekt beschreibt, sowie die Wahl der Qualitätsfunktion $\mu(\mathbf{p})$ und der Distanzfunktion $d(\mathbf{p}, \mathbf{q})$. Darüber hinaus ist es auch möglich mittels dieser Akkumulation nicht nur Objekte, sondern auch Transformationen innerhalb einer Punktmenge zu bestimmen. Dies wird in Abschnitt 5.2.4 genutzt, um Extrusionsfeatures zu finden.

5.2.2 Quader

Achsenparallele Quader können durch sechs Parameter

$$\mathbf{p} = (x_{min}, x_{max}, y_{min}, y_{max}, z_{min}, z_{max})$$

beschrieben werden. Wir suchen hier nach Quadern, deren Kanten in der Voxelstruktur enthalten sind. Um dies zu erreichen, definieren wir die Qualitätsfunktion $\mu(\mathbf{q})$ in geeigneter Weise: Für einen gegebenen Quader ordnen wir entlang jeder Kante eine Anzahl von Zellen (hier zehn) an, wobei das Zentrum der ersten bzw. letzten Zelle mit dem Anfangs- bzw. Endpunkt der Kante zusammenfällt, und überprüfen, wieviel Prozent all dieser Zellen Voxel enthalten (Abb. 24). Die Ausdehnung einer Zelle entlang der Kantenrichtung wird durch die Kantenlänge und die Zellenanzahl bestimmt, für die beiden übrigen Richtungen beträgt hier die Breite 7 Voxel.

Als Distanz zwischen zwei Tupeln verwenden wir den maximalen Abstand korrespondierender Parameter:

$$d(\mathbf{p}, \mathbf{q}) = \max_{i=1,\ldots,6} |p_i - q_i|.$$

Um die sechs Parameter zu bestimmen wählen wir zufällig n Punkte und bestimmen deren umschreibenden Quader. Die Wahl von n beeinflußt dabei

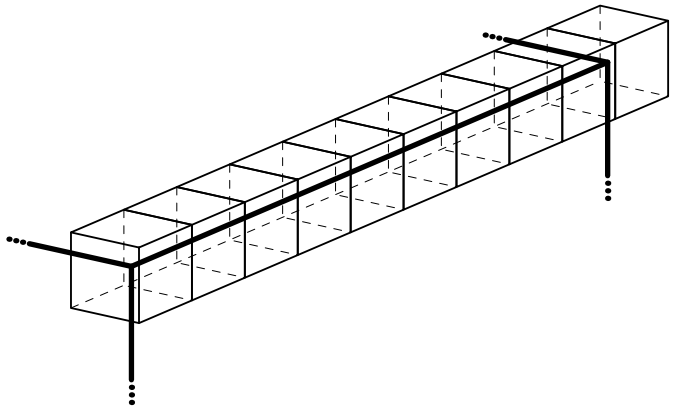


Abbildung 24: Anordnung von Zellen entlang einer Quaderkante.

die Erkennungsqualität und sollte von der Beschaffenheit der Daten abhängig sein. Um die Chance zu erhöhen, den richtigen Quader zu finden, sollte in einer Punktmenge, die nur aus einem Quader besteht, eine größere Zahl von Punkten gewählt werden. Wenn aber neben dem Quader noch zahlreiche andere Störpunkte vorhanden sind, die nicht zum Quader gehören, sollte ein kleineres n gewählt werden. Durch ein größeres n würde auch das Risiko steigen, einige der Störpunkte auszuwählen, was wiederum die Chance verringert, den korrekten Quader zu finden. Auf eine detaillierte Analyse dieses Sachverhalts wird hier verzichtet, stattdessen wurden durch ein Experiment geeignete Werte für n bestimmt.

Tab. 1 zeigt das Ergebnis eines einfachen Tests: In einem Voxelbereich der Größe $100 \times 100 \times 100$ wird zufällig jeweils ein Quader mit 50 Punkten auf jeder Kante erzeugt und gegebenenfalls werden noch einige Störpunkte hinzugefügt. Anschließend werden für alle Werte $n = 2 \ldots 18$ n Punkte gewählt, deren umschreibender Quader bestimmt und es wird überprüft, ob er mit dem zuvor erzeugten Quader übereinstimmt. Diese Prozedur wird für jeden Wert von n 1000-mal ausgeführt und alles insgesamt 1000-mal wiederholt. In der Tabelle ist für jedes n die Trefferquote in Prozent angegeben, in den Spaltenüberschriften steht dabei die Anzahl der hinzugefügten Störpunkte. Der Maximalwert ist in jeder Spalte mit einem $*$ markiert. Wie erwartet steigt die Trefferqualität mit n an, wenn keine Störungen vorhanden sind. Falls Störpunkte da sind, erhalten wir gute Resultate durch Wahl eines kleinen n, im folgenden wird immer $n = 6$ verwendet.

	$N = 0$	150	300	600	1200
$n = 2$	0.23	0.14	0.12	0.027	0.012
3	5.59	2.83	1.69	0.70	0.19
4	22.14	9.14	4.41	$*$ 1.39	$*$ 0.27
5	41.27	13.87	$*$ 5.72	$*$ 1.39	0.20
6	58.50	$*$ 15.94	5.53	1.01	0.093
7	71.53	15.78	4.55	0.66	0.050
8	80.70	14.43	3.56	0.41	0.025
9	87.28	12.56	2.55	0.23	< 0.001
10	91.51	10.64	1.86	0.13	< 0.001
11	94.44	8.90	1.31	0.078	< 0.001
12	96.31	7.20	0.92	0.039	< 0.001
13	97.57	5.91	0.65	0.024	< 0.001
14	98.41	4.84	0.42	< 0.001	< 0.001
15	98.95	3.89	0.30	< 0.001	< 0.001
16	99.31	3.19	0.19	< 0.001	< 0.001
17	99.55	2.61	0.14	< 0.001	< 0.001
18	$*$ 99.71	2.04	0.09	< 0.001	< 0.001

Tabelle 1: Trefferquote für einen Quader mit 50 Punkten auf jeder Kante und zusätzlichen N Störpunkten bei Auswahl von n Punkten.

Außerdem ist die korrekte Erkennung mehrerer Objekte ein weiteres Problem. In einer Szene mit k Quadern ist die Wahrscheinlichkeit dafür, n Punkte des gleichen Quaders auszuwählen, gleich $\left(\frac{1}{k}\right)^{n-1}$ (falls ein Punkt mehrmals gewählt werden kann). Dieser Wert verringert sich mit zunehmendem k. Dennoch kann diese Methode erfolgreich für unsere Voxeldaten angewandt werden, indem wir nicht versuchen alle Quader automatisch zu finden, sondern dem Benutzer die Möglichkeit geben, den Erkennungsprozess zu steuern. In unserem Softwareprototyp ist es möglich, eine Region von Voxeln für die Erkennung auszuwählen. Nach dem Akkumulationsprozess werden die gefundenen Quader dem Benutzer präsentiert und er kann den gewünschten Quader daraus auswählen. Abb. 25 zeigt ein Beispiel. Die Parameter $step_{max}$, μ_{min}, d_{min} sind dabei vom jeweiligen Modell abhängig und können vom Benutzer eingestellt werden.

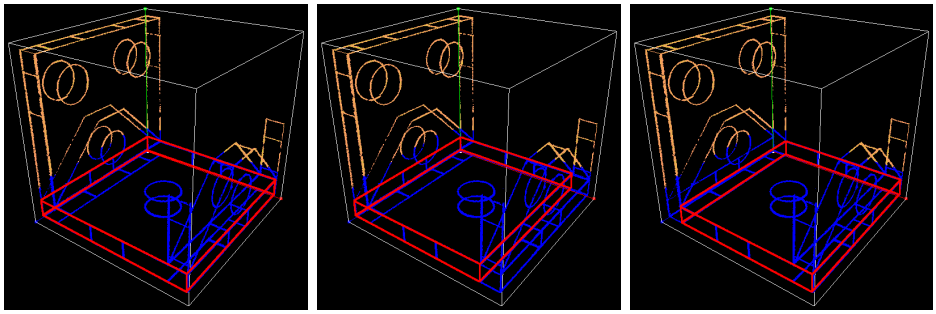


Abbildung 25: Drei erkannte Quader (rot) in einer ausgewählten Teilmenge des Voxelmodells.

5.2.3 Zylinder

Achsenparallele Zylinder sind eine weitere Objektklasse, die mit der beschriebenen Akkumulationstechnik gefunden werden können. Sie erscheinen in der Voxelstruktur als zwei Kreise auf parallelen Ebenen (vorausgesetzt die Begrenzungsflächen des Teils verlaufen achsenparallel). Um sie zu beschreiben, verwenden wir sechs Parameter: Die Koordinatenrichtung, in der die Achse des Zylinders verläuft (p_1), zwei Koordinaten, die die Lage der Achse auf der Ebene senkrecht zu ihr definieren (p_2, p_3), den Radius (p_4) und zwei Werte für den Koordinatenbereich, die der Zylinder entlang der Achse einnimmt. Der Algorithmus muss dabei leicht modifiziert werden, um den ersten Parameter korrekt zu behandeln, da er lediglich ein Flag darstellt. Durch die Distanzfunktion muss sichergestellt werden, dass Tupel mit verschiedenen Richtungen nie im gleichen Cluster akkumuliert werden. Die Differenz zwischen zwei Parametertupeln $\mathbf{p} = (p_1, \ldots, p_6)$ und $\mathbf{q} = (q_1, \ldots, q_6)$ wird definiert als

$$d(\mathbf{p}, \mathbf{q}) = \begin{cases} \infty & \text{falls } p_1 \neq q_1 \\ \max\limits_{i=2,\ldots,6} |p_i - q_i| & \text{falls } p_1 = q_1 \end{cases} .$$

Zur Bestimmung eines Parametertupels wählen wir drei Punkte und berechnen ein mögliches Tupel für jede der drei Achsenrichtungen (dies entspricht der Bestimmung eines Kreises durch drei Punkte in der Ebene). Als Koordinatenbereich verwenden wir die entsprechenden minimalen bzw. maximalen Koordinaten der drei Punkte.

Um die Qualität eines Parametertupels zu bestimmen ordnen wir analog zur Qualitätsfunktion für Quader entlang der beiden Kreise Zellen an (hier 20 pro Kreis) und zählen, wieviel Prozent von ihnen Punkte enthalten.

5.2.4 Extrusionen

Neben dem Finden von Objekten kann die Akkumulation auch dazu genutzt werden, Transformationen innerhalb von Punktmengen zu identifizieren. Beispiele dafür zeigt die Abb. 26. Links oben ist eine Punktmenge sowie ihr Bild bei einer Translation (beide hervorgehoben) und einige zusätzliche Störpunkte zu sehen. Um herauszufinden, ob in dieser Gesamtmenge zwei Teilmengen enthalten sind, die durch eine Translation aufeinander abgebildet werden können, werden alle Paare von paarweise verschiedenen Punkten betrachtet und die zugehörige Translation in einem Akkumulator vermerkt. Dabei entsteht die rechts oben abgebildete Struktur, in der sich deutlich zwei Peaks abzeichnen, die die zwei möglichen Translationen beschreiben. Schwierigkeiten treten allerdings auf, wenn im Bild regelmäßige Strukturen enthalten sind, wie etwa die Rechtecke in der unteren Reihe. Zusätzlich zu den beiden eigentlichen Peaks können sich dann noch weitere Gebiete mit hohen Akkumulatorwerten herausbilden, was unter Umständen zu Problemen führt. Dem kann jedoch durch Verwendung von a-priori-Wissen begegnet werden. Neben Translationen können mit dieser Methode auch andere Transformationen detektiert werden.

Damit können nun in der Voxelstruktur z. B. Extrusionen, d. h. Features, die durch Verschieben eines Profils entstanden sind, gefunden werden. Um die entsprechenden Translationen zu finden, wird ein dreidimensionaler Akkumulator benutzt und es wird für jedes Paar von zwei verschiedenen Punkten die entsprechende Translation akkumuliert (die Punktepaare werden hier nicht zufällig ausgewählt). Enthält nun die Voxelstruktur Profile, die durch eine Translation verbunden sind, sollten sich auch hier dazugehörige Peaks im Akkumulator abzeichnen.

Dabei gibt es auch hier das oben erwähnte Problem mit regelmäßigen Strukturen: Die im Voxelmodell enthaltenen geraden Kanten würden ebenfalls Peaks produzieren, da sich auf ihnen zahlreiche Paare von Punkten mit gleichem Abstand finden lassen. Um dem zu begegnen, wird vor der Akkumulation einer Translation geprüft, ob sich entlang der Verbindungslinie der

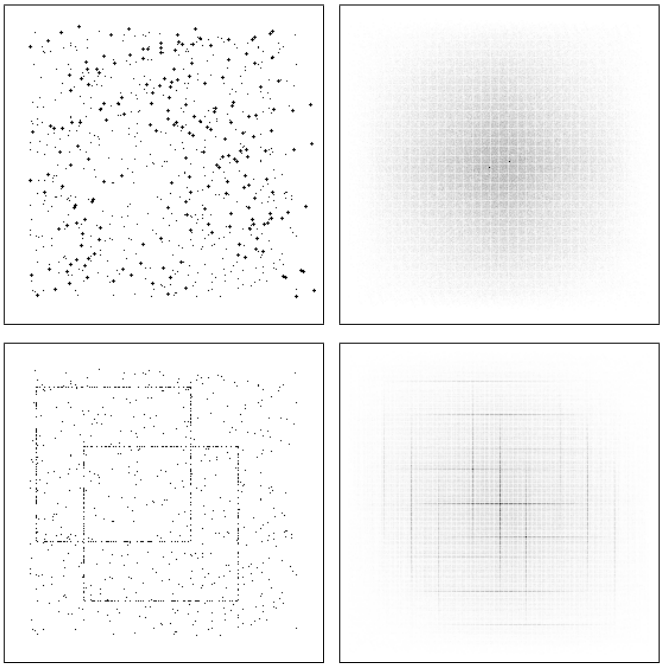


Abbildung 26: Ausgangsmenge und Akkumulatorinhalt für die Suche nach Translationen, s. Abschnitt 5.2.4.

Punkte zahlreiche andere Punkte befinden. Ist dem so, wird diese Translation nicht akkumuliert. Gegebenenfalls kann in einem Vorverarbeitungsschritt eine Datenstruktur aufgebaut werden, die einen solchen schnellen Test möglich macht.

Die oben beschriebene Listenakkumulation ist in diesem Fall aufgrund der großen Menge entstehender Cluster nicht unbedingt geeignet.

Abb. 27 zeigt den Akkumulatorinhalt für eine achsenparallele Box, wobei die Größe der kleinen Würfel die Anzahl der Elemente im jeweiligen Cluster veranschaulichen soll. Wie erwartet erhält man sechs Peaks für die sechs Verschiebungen, die jeweils eine Seitenfläche auf die ihr gegenüberliegende abbilden. Da allerdings beliebige Translationen zugelassen sind, werden auch zahlreiche andere akkumuliert, die dann "schrägen" Verschiebungen entsprechen.

In Abb. 28 ist das Ergebnis bei Anwendung auf das bisherige Beispielteil zu sehen (eine Translation ist durch eine verschiedenfarbige Ausgangs-

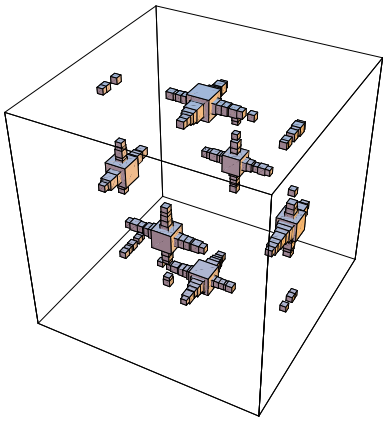


Abbildung 27: Akkumulatorinhalt für eine achsenparallele Box.

und Bildmenge gekennzeichnet). Offensichtlich werden zwei Verschiebungen gefunden, dabei ist für die Erzeugung des trapezförmigen Extrusionskörpers im Vordergrund nur die erste von Interesse.

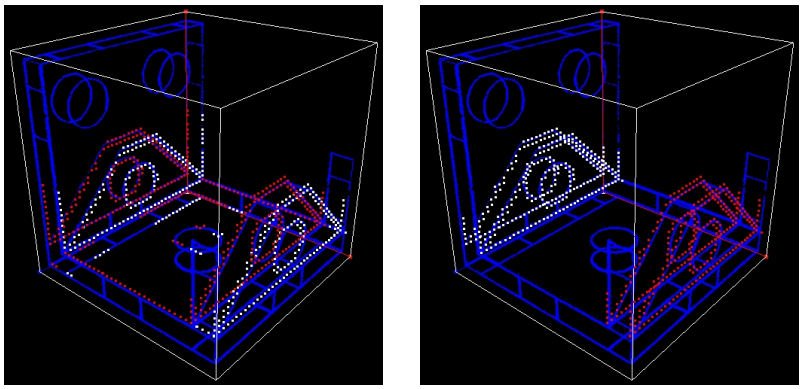


Abbildung 28: Zwei gefundene Translationen.

Um die Performance zu verbessern, ist es möglich, vor der Akkumulation das Voxelmodell in ein Gitter mit gröberer Auflösung zu übertragen, um die zusammengehörigen Punktmengen zunächst grob zu identifizieren und die korrekte Translation in einem zweiten Schritt mit der ursprünglichen Genauigkeit zu bestimmen.

Ebenso ist eine Beschränkung auf achsenparallele Translationen möglich, damit würden dann auch drei eindimensionale Akkumulatorarrays genügen.

5.3 Automatisches Finden von Features

Im vorangehenden Abschnitt 5.2 wurde beschrieben, wie in der erhaltenen Voxelstruktur mit einer Akkumulationstechnik Features gefunden werden können. Dieses Vorgehen hat allerdings den Nachteil, dass eine recht intensive Benutzerinteraktion notwendig ist, um jeweils den interessierenden Bereich auszuwählen. In diesem Abschnitt soll daher als Ergänzung dazu ein Verfahren vorgestellt werden, mit dem die Features (Quader, Zylinder) in der gegebenen Voxelstruktur weitgehend vollautomatisch identifiziert werden können, ohne dass der Benutzer eine Regionenmarkierung vornehmen muss.

5.3.1 Quader

Um Quader zu finden, wird zunächst versucht, linienartige Strukturen bzw. Segmente zu bestimmen, die als Kanten von Quadern fungieren können. Gekennzeichnet sind solche Strukturen natürlich durch eine Anhäufung nebeneinander angeordneter Voxel, und da hier ausschließlich achsenparallele Quader von Interesse sind, sollten die Koordinaten der entsprechenden Trägergeraden durch eine Projektion und Akkumulation entlang der jeweiligen Koordinatenrichtung zu ermitteln sein. Dies konnte auch im Experiment erfolgreich umgesetzt werden, allerdings hat es sich dabei als nützlich erwiesen, für ein Voxel nicht nur das Voxel selbst, sondern noch einige umliegende weitere Voxel zu akkumulieren. In der vorliegenden Lösung wird ein Block der Größe $3 \times 3 \times 5$ verwendet, so werden also z. B. bei der Suche nach in x-Richtung verlaufenden Segmenten für ein Voxel mit den Koordinaten (x, y, z) alle 45 Voxel mit Koordinaten im Bereich $[x-2, x+2] \times [y-1, y+1] \times [z-1, z+1]$ in einem zweidimensionalen, parallel zur yz-Ebene angeordneten Akkumulator erfasst (s. Abb. 29).

Um die Koordinaten für die Trägergeraden zu bestimmen, kann nun im jeweils erhaltenen Akkumulatorbild nach Peaks gesucht werden. Dies gestaltet sich jedoch schwierig, da im Bild auch zahlreiche linienartige Strukturen enthalten sind, und aufgrund der unterschiedlichen Länge der zu findenden

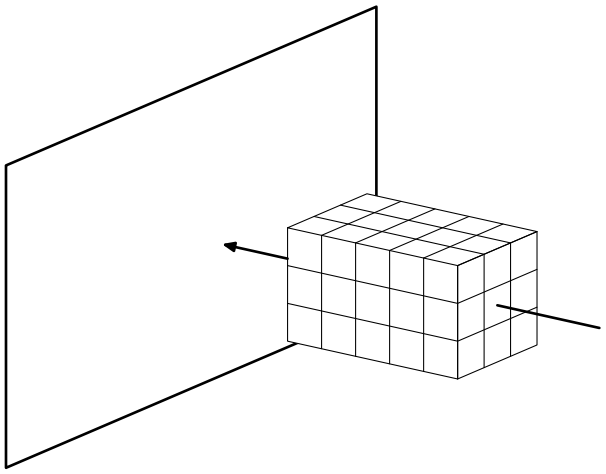


Abbildung 29: Projektion von Voxeln entlang einer Geraden in einen 2D-Akkumulator.

„projizierten Segmente“ weisen die Peaks im Bild auch sehr unterschiedliche Höhen auf. Daher wird ein geeigneter Schwellwert bestimmt, so dass bei Auslassen aller Punkte mit einer Intensität unterhalb dieses Wertes das Akkumulatorbild frei von Linien wird. Die Abb. 30 veranschaulicht diese Situation.

Um die Koordinaten der Trägergeraden zu bestimmen, werden nun die im Akkumulatorbild verbliebenen Punkte verwendet und es wird ein Clustering durchgeführt, das im Ergebnis die gesuchten Werte liefert. Damit sind die Lagen der Trägergeraden ermittelt, und um nun die tatsächlichen Kandidaten für Quaderkanten zu finden, wird eine Zellenunterteilung entlang jeder Trägergeraden vorgenommen. In jeder Zelle wird vermerkt, ob in ihr Voxel enthalten sind. Findet sich nun eine bestimmte Anzahl zusammenhängender belegter Zellen entlang einer Geraden, so wird dieser Abschnitt als ein gültiges Segment interpretiert (s. Abb. 31).

Über die tatsächliche Qualität des Voxelmodells ist natürlich an dieser Stelle wenig bekannt, daher wurde für die weiteren Schritte ein Vorgehen gewählt, das sicher nicht in jedem Fall angewendet werden kann, in den bisherigen Experimenten aber sehr zufriedenstellende Ergebnisse geliefert hat. Wir gehen hier davon aus, dass die Qualität des Modells wenigstens so gut ist, dass sich in jedem darin enthaltenen Quader wenigstens an einem Eckpunkt und den drei dazu benachbarten Eckpunkten zwei Segmente bzw.

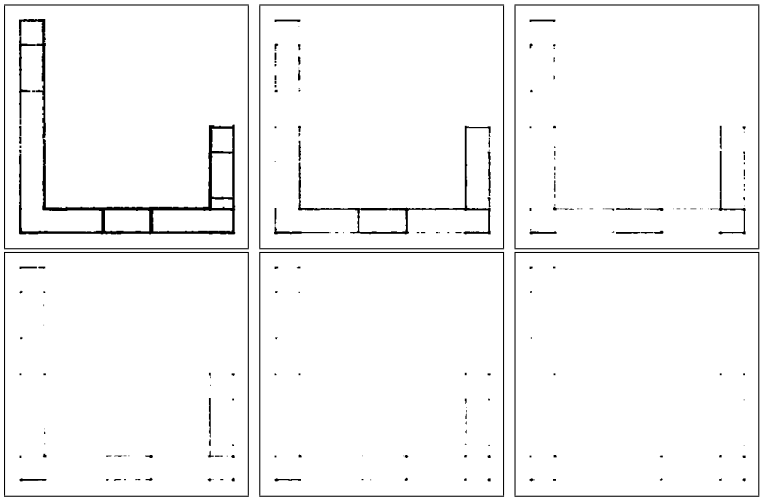


Abbildung 30: 2D-Akkumulatorbild (links oben) sowie bei Berücksichtigung verschiedener Schwellwerte entstehende Bilder.

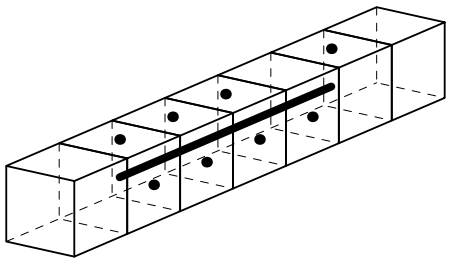


Abbildung 31: Finden der Segmente entlang einer Trägergeraden.

Kantenfragmente schneiden. Unter dieser Voraussetzung können wir nun alle Schnittpunkte zwischen jeweils zwei Segmenten verschiedener Richtung bestimmen. Anschließend wird für jeden Schnittpunkt dieser Menge geprüft, ob es für ihn in jeder der drei Koordinatenrichtungen einen weiteren gibt, der mit ihm (innerhalb einer vorgegebenen möglichen Abweichung) auf einer Geraden liegt. Werden derartige vier Punkte gefunden, werden sie als mögliche Kandidaten für Ecken eines Quaders betrachtet, und es wird geprüft, wie gut der von ihnen aufgespannte Quader im Voxelmodell repräsentiert wird (s. Abb. 32). Dazu wird wieder die in Abschnitt 5.2.2 beschriebene Qualitätsfunktion genutzt, die eine Zellenunterteilung entlang der Quaderkanten vornimmt und die prozentuale Belegung als Güte verwendet.

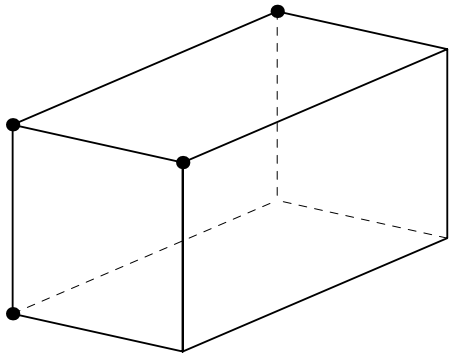


Abbildung 32: Von 4 Punkten aufgespannter Quader.

Anschließend wird noch ein Clustering für die erhaltenen Parametersätze der Quader durchgeführt, um möglicherweise entstandene sehr nah beieinanderliegende Lösungen innerhalb eines bestimmten Genauigkeitsbereichs zu einer zu fusionieren. Eine Voxelstruktur mit dem dabei erzielten Erkennungsergebnis wird in den Abbn. 43 und 44 auf den Seiten 59 und 60 gezeigt.

5.3.2 Zylinder

Im Anschluss an die Detektion von Quadern kann in einem zweiten Schritt die Detektion von Zylindern vorgenommen werden. Diese Reihenfolge bietet sich an, da die Zylinderdetektion erheblich vereinfacht wird, wenn zuvor alle Voxel, die gewissermaßen schon für Quader verbraucht wurden, aus dem Modell entfernt werden. Da die Zylinder im Voxelmodell durch zwei Kreise mit annähernd gleichem Radius, deren Mittelpunkte (ebenfalls näherungsweise) auf einer gemeinsamen achsenparallelen Geraden liegen, repräsentiert werden, ist es zunächst günstig, solche kreisförmigen Strukturen, die in den meisten Fällen auch klar regional von übrigen Elementen des Modells abgegrenzt sind, zu identifizieren.

Dazu wird nach dem Entfernen der zu den gefundenen Quadern gehörenden Voxel eine Suche nach zusammenhängenden Komponenten im Voxelmodell vorgenommen. Der Zusammenhang zwischen zwei Voxeln wird dabei durch einen geeignet vorgegebenen Abstand definiert. In der vorliegenden Implementierung werden dazu die Voxel als Knoten eines Graphen interpretiert, und in einem ersten Schritt werden alle Voxel, deren Abstand kleiner

oder gleich dem vorgegebenen Abstand ist, durch eine Kante verbunden. In diesem Graphen werden nun durch wiederholtes Auswählen eines Knotens, einer Tiefensuche ausgehend von diesem Knoten und anschließendes Entfernen der gefundenen Knoten aus dem Graphen und Zusammenfassen zu einem eigenständigen Graphen Schritt für Schritt alle Zusammenhangskomponenten gefunden.

In jeder der Zusammenhangskomponenten wird nach Kreisen gesucht, dies geschieht mittels der in Abschnitt 5.2.3 beschriebenen Akkumulationstechnik für Zylinder. Um anstelle von Zylindern Kreise zu erhalten, wird die Qualitätsfunktion so modifiziert, dass nur Parametertupel $(p_1, \ldots, p_6)$ akzeptiert werden, für die die Länge des Koordinatenbereichs entlang der jeweiligen Koordinatenrichtung kleiner als ein geeignet vorgegebener Wert (hier 5) ist.

Alle in den Zusammenhangskomponenten gefundenen Kreise werden in drei Mengen aufgeteilt, die den drei Koordinatenrichtungen entsprechen. In jeder der drei Mengen werden dann Cluster gebildet, wobei in einem Cluster alle die Kreise zusammengefasst werden, deren Mittelpunkte auf einer Geraden liegen und deren Radien übereinstimmen (d. h. innerhalb einer vorgegebenen Grenze ähnlich sind, da eine exakte Übereinstimmung hier sicher nicht erreicht werden kann). Anschließend erfolgt noch eine Tiefensortierung der Kreise eines Clusters, und jeder Bereich zwischen zwei gemäß dieser Sortierung benachbarten Kreisen ist ein Kandidat für ein Zylinder-Feature, welches dann anschließend weiterverarbeitet werden kann.

5.4 Teilegenerierung

Die beiden Abschnitte 5.2 und 5.3 haben gezeigt, wie in der Voxelstruktur Quader und Zylinder als Features bzw. Grundbausteine für das dargestellte CAD-Teil gefunden werden können. In diesem und im folgenden Abschnitt soll nun gezeigt werden, wie diese gefundenen Elemente zu kompletten Teilen kombiniert werden können, um eine Liste von Kandidaten für das gesuchte Teil zu finden, und wie diese Kandidaten anschließend bewertet werden können.

Die grundlegende Vorgehensweise dabei ist, die gefundenen Features auf alle möglichen Arten mittels der Booleschen Operationen Vereinigung, Durchschnitt und Differenz zu kombinieren, von jedem so entstandenen Teil die

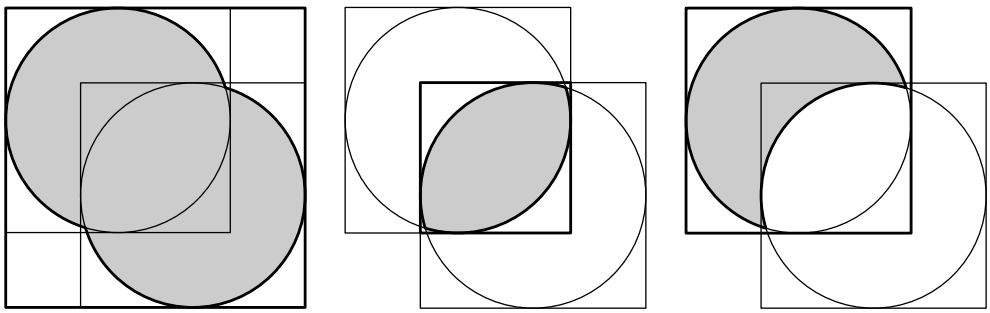


Abbildung 33: Die Booleschen Operationen Vereinigung, Durchschnitt und Differenz sowie die auf der Basis der umschreibenden Rechtecke der Objekte berechneten umschreibenden Rechtecke für das Ergebnis der jeweiligen Operation.

zu den eingescannten Projektionen gehörenden 2D-Projektionen zu erzeugen und beide miteinander auf ihre Ähnlichkeit hin zu überprüfen. Die Booleschen Operationen werden in Abb. 33 für den zweidimensionalen Fall veranschaulicht. Von Interesse ist dabei besonders die Berechnung der umschreibenden Boxen der Objekte, da diese im unten beschriebenen Algorithmus verwendet werden, um bereits vorab ohne tatsächliches Ausführen der Booleschen Operationen den Suchaufwand durch Beachten geometrischer Lagebeziehungen erheblich einzuschränken.

Die entstehenden Körper werden hier durch binäre Bäume repräsentiert. Der einfachste Fall ist dabei natürlich ein Baum mit einem einzelnen Knoten, der ein einzelnes Feature repräsentiert, für komplexere Teile repräsentieren die Blätter des Baums die Features, die für das Teil verwendet werden, während die inneren Knoten die zu verwendenden Booleschen Operationen beinhalten. Die Abb. 34 zeigt ein Beispiel.

Im folgenden wird eine rekursive Prozedur `GenerateParts()` angegeben, die zwei Argumente besitzt: Zum einen eine Menge von Elementen $E = \{e_1, \ldots, e_n\}$, die die Geometrieelemente enthält, die durch die Prozedur weiter kombiniert werden sollen, um letztendlich gültige Teile zu erzeugen, zum anderen die zu Beginn leere Menge P, die alle während des Verfahrens entstehenden Teile aufnimmt. Unter der *Featuremenge* $F(e)$ wird hier die Menge aller Features verstanden, die für das Element e verwendet werden, d. h., die durch die Blätter des zu e gehörenden binären Baums repräsentiert werden. Als Randbedingung für die Teileerzeugung wird hier festgelegt,

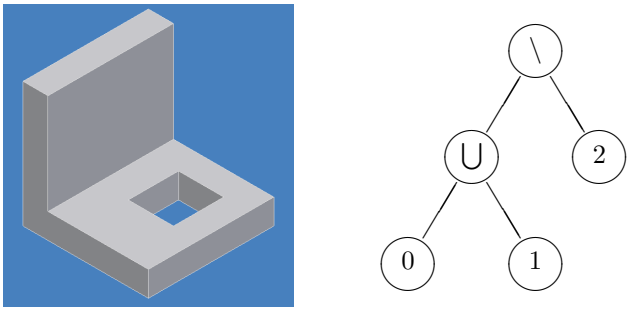


Abbildung 34: Ein einfaches Teil sowie seine Repräsentation als binärer Baum (0 – aufrecht stehender Quader, 1 – waagerecht liegender Quader, 2 – von 1 entfernter Quader).

dass jedes Feature nur genau einmal in den zu erzeugenden Teilen verwendet werden darf.

`GenerateParts(`$E = \{e_1, \ldots, e_n\}$`, `P`)`

1. Enthält E genau ein Element, dieses als mögliches Teil zu P hinzufügen, Prozedur beenden.
2. Eine Menge $C = \emptyset$ für zu bildende Kombinationen anlegen.
3. Für jedes Paar $(e_i, e_j) \in E \times E$ mit $i < j$ prüfen, ob sich die umschreibenden Boxen $B(e_i)$ und $B(e_j)$ der beiden Elemente schneiden und ob die Featuremengen $F(e_i)$ und $F(e_j)$ disjunkt sind. Ist beides der Fall, die entsprechende Kombination (e_i, e_j) in C aufnehmen.
4. Für jede nichtleere Teilmenge $C' = \{c_1, \ldots, c_m\}$ von C:
   (a) Prüfen, ob ein Element e_i in zwei Kombinationen aus C' verwendet wird. Ist dies der Fall, mit der nächsten Teilmenge bei 4 fortfahren.
   (b) Alle Elemente aus E, die nicht in einer Kombination aus C' benutzt werden, in der Menge E' zusammenfassen.
   (c) Für jede Kombination $c_i = (e_p, e_q)$ aus C' ein neues Element bilden, indem die beiden Elemente e_p und e_q durch einen neuen Knoten zu einem binären Baum t_i verknüpft werden.

(d) Für jedes Element aus $\{\cup, \cap, \setminus\}^m$ die Wurzeln der neu erzeugten m Bäume mit der entsprechenden Operation belegen, der dabei entstehende Baum sei mit t'_i bezeichnet. Dabei sind aufgrund der Nichtkommutativität der Differenz $\setminus$ die beiden Teilbäume einmal in der Originalreihenfolge, einmal vertauscht zu verwenden, so dass für jeden in Schritt 4c erzeugten Baum insgesamt vier mögliche Varianten entstehen. Für jede dieser möglichen Belegungen `GenerateParts(`$E' \cup \{t'_1, \ldots, t'_m\}$`, P)` rekursiv aufrufen.

Das Terminieren der Rekursion ist dabei sichergestellt, denn ein rekursiver Aufruf erfolgt nur dann, wenn die Menge der Kombinationen nicht leer ist. Das bedeutet aber, dass wenigstens zwei Elemente zu einem neuen zusammengefasst werden, so dass der rekursive Aufruf immer mit wenigstens einem Element in E weniger erfolgt, was zwangsläufig irgendwann zum Abbruch der Rekursion führt.

Ein Beispiel für die entstehenden möglichen Teile zeigt die Abb. 35. Als Primitive wurden hier die drei Quader benutzt, aus denen das Teil in Abb. 34 gebildet wird. In diesem Fall liefert die Prozedur `GenerateParts()` insgesamt 16 mögliche Bäume bzw. Teile.

`GenerateParts()` ist natürlich geeignet, alle möglichen Teile zu generieren. Allerdings zeigt schon das im letzten Absatz erwähnte Beispiel, dass in dieser allgemeinen Form eine sehr große Anzahl von möglichen Kandidaten generiert wird.

Deshalb ist es erforderlich, durch die Berücksichtigung weiterer geometrischer Sachverhalte und Lagebeziehungen den Suchaufwand zusätzlich zu begrenzen. Als wesentliches Element für diese Betrachtungen werden dazu wiederum die umschreibenden Boxen der Primitive und der aus ihnen kombinierten Teile herangezogen.

Als Vorverarbeitungsschritt wird ein Graph konstruiert, dessen Knoten die Primitive repräsentieren. Zwei Knoten werden durch eine Kante genau dann verbunden, wenn der Durchschnitt der umschreibenden Boxen der beiden den Knoten zugeordneten Primitive nichtleer ist.

Unter der Voraussetzung, dass jedes Primitiv im Teil zu verwenden ist und dass das gesuchte Teil zusammenhängend sein muss, ist sofort ersichtlich, dass Primitive, für die ein Entfernen ihres Knotens und der mit ihm

inzidenten Kanten aus dem Graphen einen Zerfall des Graphen in mehrere Zusammenhangskomponenten zur Folge hat, auf jeden Fall einen „positiven“ Beitrag an Material zum Teil liefern müssen. Solche Primitive seien im folgenden *Brückenelemente* genannt.

Weiterhin ist klar, dass zwei Brückenelemente, deren Knoten im Graphen direkt benachbart sind, nur durch eine Vereinigung miteinander kombiniert werden können. Eine Kombination mittels Durchschnitt oder Differenz würde aufgrund der Eigenschaft eines Brückenelements zu einem nicht zusammenhängenden Teil bzw. zu einem Abbruch der Rekursion in der Prozedur `GenerateParts()` führen. Allerdings muss diese Vereinigung nicht direkt geschehen, sondern es ist möglich, dass mehrere benachbarte Brückenelemente zu vereinigen sind. Dies kann auf einfache Weise durch Verwenden einer leicht modifizierten Version von `GenerateParts()` geschehen, indem alle Brückenelemente einer solchen aus benachbarten Elementen bestehenden Menge als Eingabe verwendet werden, als Operation nur die Vereinigung zugelassen wird und die Rekursion nach dem Erhalt der ersten Lösung abgebrochen wird.

Auf diese Weise werden zusammenhängende Mengen von Brückenelementen in einem Vorverarbeitungsschritt zu jeweils einem neuen Element zusammengefasst und gehen als ein Element in die Prozedur `GenerateParts()` ein.

Zuvor lässt sich jedoch noch ein weiterer Schritt ausführen: Nachdem die Brückenelemente identifiziert wurden, wird geprüft, ob es Elemente gibt, deren Knoten nur durch eine einzige Kante mit dem Knoten des Brückenelements verbunden sind, und deren umschreibende Box eine Teilmenge der Box des Brückeneelements ist. Zwischen einem solchen Element und seinem zugehörigen Brückenelement ist nur eine Kombination mittels Differenz sinnvoll, wobei das Brückenelement der erste Operand ist. Eine solche Situation findet man z.B. bei einer Bohrung in einer Platte. Dieser Test zusammen mit dem Verbinden der entsprechenden Elemente zu einem neuen, welches dann anstelle des Brückenelements weiterverwendet wird, wird direkt nach der Identifikation der Brückenelemente und vor der Kombination zusammenhängender Brückenelemente ausgeführt.

Neben diesen Optimierungen vor dem Einsatz von `GenerateParts()` lässt sich auch noch in der Prozedur selbst eine Verbesserung vornehmen. So kann vor Schritt 3 geprüft werden, ob es Elemente gibt, die jeweils nur mit genau

einem anderen einen nichtleeren Durchschnitt ihrer umschreibenden Boxen haben. Ist das der Fall, wird ein solches Element beliebig ausgewählt und gemeinsam mit seinem Gegenpart als einzige Kombination in die Menge C aufgenommen. Das Verfahren wird dann unter Auslassen von Schritt 3 direkt mit Schritt 4 fortgeführt.

Zum Abschluss ist es noch sinnvoll, mit einem einfachen Test die von `GenerateParts()` erzeugten Teile zu überprüfen, ob sie mögliche Lösungen darstellen können. Dazu wird jeweils die umschreibende Box des Teils betrachtet (sie wird während der Teilegenerierung berechnet und mitgeführt). Ganz offensichtlich darf es nicht vorkommen, dass sich die umschreibende Box eines Brückenelements außerhalb der Teile-Box befindet, da sonst dieses Brückenelement keinen Einfluss auf das Teil mehr hätte. Teile, bei denen das vorkommt, werden als Kandidaten verworfen.

5.5 Teilebewertung

Die im vorhergehenden Abschnitt erzeugten Bäume können nun weiter daraufhin überprüft werden, ob sie erstens gültige Teile beschreiben und wenn ja, ob zweitens das jeweilige Teil zu den vorgegebenen Projektionen passt und damit als mögliche Lösung infrage kommt. Dazu ist es notwendig, die in den Bäumen festgelegten Booleschen Operationen tatsächlich auszuführen und anschließend vom entstandenen Teil die 2D-Projektionen abzuleiten. In der vorliegenden Implementierung wurde dazu die Software OpenCASCADE [47] genutzt. Sie bietet einen Geometriemodeller, der zur Entwicklung eigener Anwendungen genutzt werden kann, und insbesondere besteht die Möglichkeit, die Ableitung der Projektionen auszuführen. Dies ist in einem Screenshot in Abb. 36 kurz dargestellt.

Die Vorgehensweise in der Implementierung ist dabei folgende: Für jeden von `GenerateParts()` erzeugten Baum wird seine Struktur mittels eines dafür definierten Textformats an den Geometriemodeller übertragen. Dort wird diese Teilebeschreibung gelesen, die Geometrie wird erzeugt, und es werden die Projektionen abgeleitet, die anschließend korrekt positioniert (gegenüber den Ausgangsprojektionen) in Bildern abgespeichert werden. Zwischen den Bildern mit den ursprünglichen gescannten Projektionen und den vom Modell abgeleiteten Projektionen werden Differenzbilder erzeugt, und

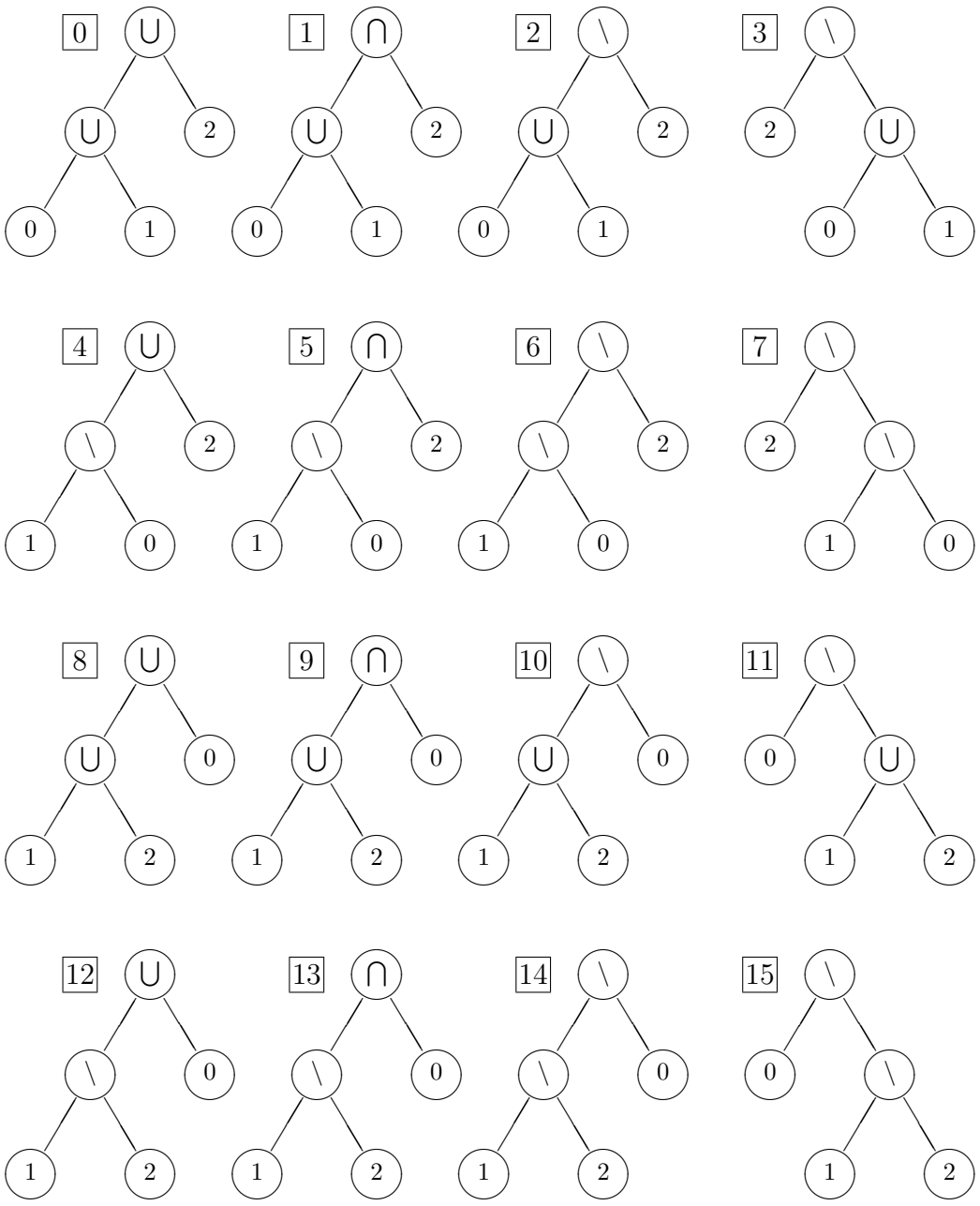


Abbildung 35: Bäume der Booleschen Operationen für die 16 aus den drei Quadern der Abb. 34 gebildeten möglichen Teile.

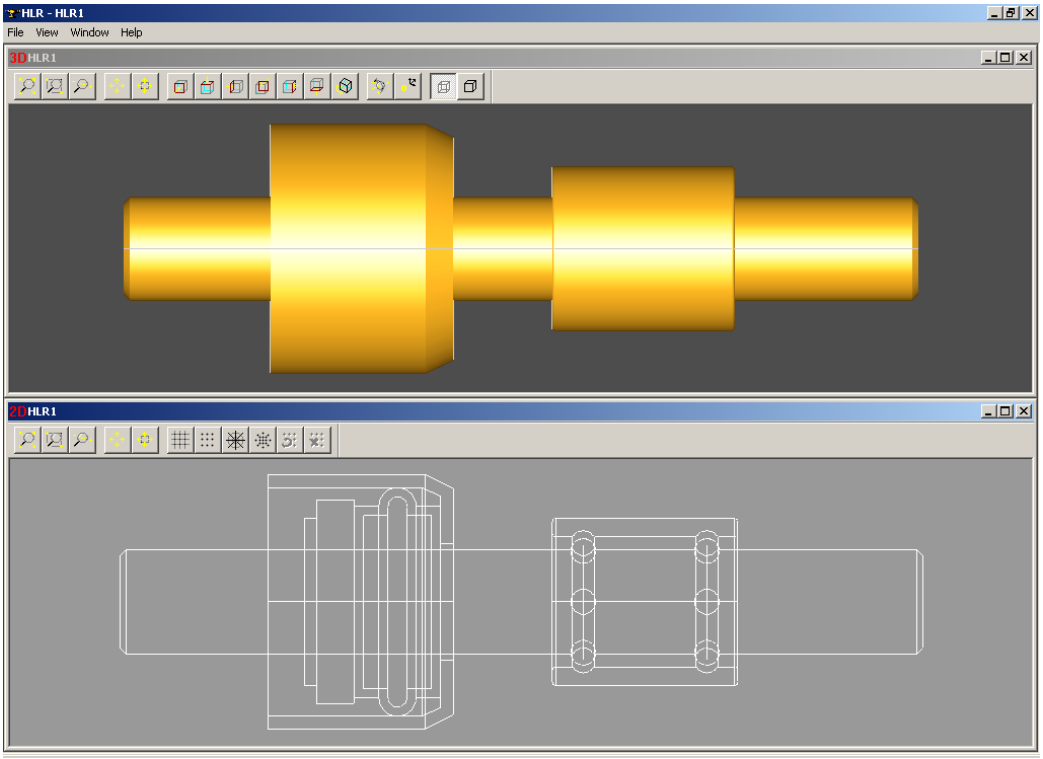


Abbildung 36: Die Vorderansicht eines 3D-Teils und die unter Verwendung von OpenCASCADE davon abgeleitete Projektion.

die darin noch verbliebenen Elemente können dann als Maß für die Übereinstimmung gewertet werden. Dies geschieht, indem die Elemente in jeweils einem Bild durch eine pixelweise Dilatation leicht verstärkt werden, und die korrespondierenden Pixel aus dem anderen Bild gelöscht werden. Vorgenommen wird das in beide Richtungen, d.h. es werden einmal die erzeugten Projektionen von den gescannten abgezogen und ebenso umgekehrt. Im Idealfall, das heißt wenn das modellierte Teil exakt dasjenige ist, welches in den gescannten Projektionen dargestellt ist, sind die Differenzbilder völlig leer.

Ein Beispiel dafür zeigen die Abbn. 37 und 38. Abb. 37 stellt die drei Projektionen des Teils aus Abb. 34 dar. In Abb. 38 sind dann für jedes der 16 Teile aus Abb. 35 die abgeleiteten Projektionen (Spalten 1 bis 3) sowie die entstehenden Differenzbilder (Spalten 4-6 für Original abzgl. Projektion, Spalten 7-9 für Projektion abzgl. Original) zu sehen.

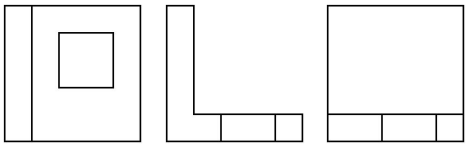


Abbildung 37: Projektionen des Teils aus Abb. 34

Als Maß für die Güte eines Differenzbildes wurde hier für ein Einzelbild einfach das Verhältnis von schwarzen Pixeln zur Gesamtpixelzahl verwendet, und um das Teil komplett zu bewerten wurde die Summe über alle sechs zugehörigen Differenzbilder gebildet. Für das angegebene Beispiel zeigt die Tab. 2 die Ergebnisse. Offensichtlich gibt es zwei Teile mit einem identischen Minimalwert (mit * markiert), die als beste Lösung infrage kommen.

Leider hat sich bei der Realisierung dieser Lösung gezeigt, dass der Algorithmus zur Ableitung der 2D-Projektionen in OpenCASCADE momentan noch nicht vollständig korrekt arbeitet (dies wurde durch den Support auch bestätigt). Aufgrund dessen ist in Spalte 1 in Abb. 38 bei den Teilen 0, 4, 8 und 10 noch die Projektion von Quader 2 zu sehen. Außerdem sind bei den Teilen 0, 2, 8 und 12 in Spalte 2 an der Stelle, an der der waagerecht liegende Quader 1 und der aufrecht stehende Quader 0 zusammentreffen, noch zwei zusätzliche und fälschlicherweise erzeugte Kanten sichtbar. Auf eine manuelle Korrektur wurde hier verzichtet, in diesem Beispiel wird die korrekte Identifikation des passenden Teils dadurch allerdings auch nicht beeinflusst.

5.6 Anwendungsbeispiel

Nach der Vorstellung der verschiedenen eingesetzten Techniken sollen in diesem Abschnitt ihre Anwendung und ihr Zusammenwirken anhand eines etwas komplexeren Beispiels demonstriert werden. Daneben werden noch zwei zusätzliche Komponenten vorgestellt, die für den praktischen Einsatz unabdingbar sind.

Die Aufgabe soll hier darin bestehen, das in Abb. 39 gezeigte Teil aus seinen Projektionen (Abb. 40) zu rekonstruieren. Dazu wird zunächst gemäß dem Vorgehen in Abschnitt 5.1 ein Voxelmodell erzeugt, welches in Abb. 41 zu sehen ist. Hier wird auch schon ein weiteres Problem deutlich: Die Teile-

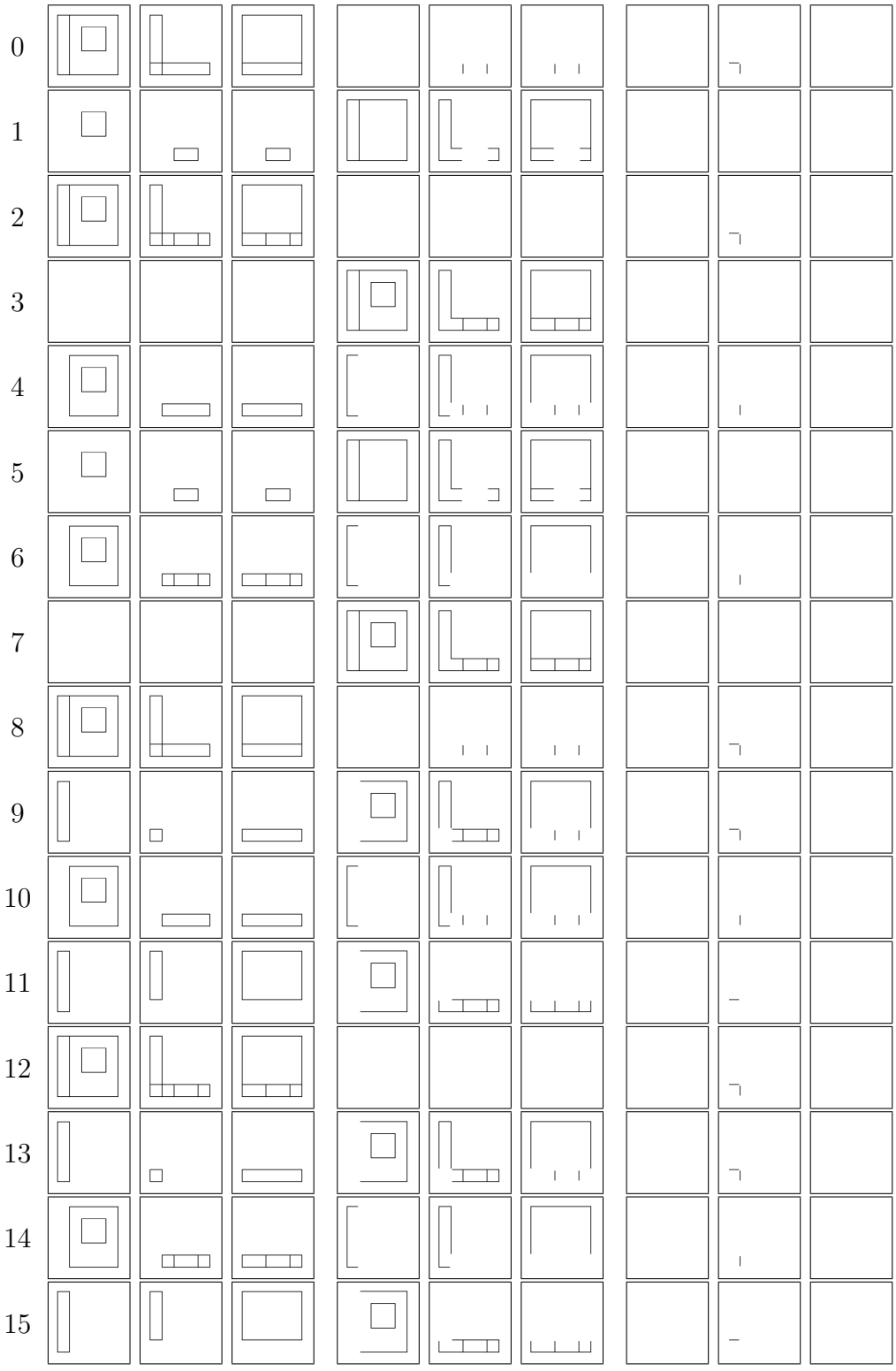


Abbildung 38: Abgeleitete Projektionen und Differenzbilder.

	Original\Projektion			Summe	Projektion\Original			Summe	Gesamt
0	0.0000	0.0031	0.0031	0.0062	0.0000	0.0031	0.0000	0.0031	0.0094
1	0.0477	0.0299	0.0394	0.1170	0.0000	0.0000	0.0000	0.0000	0.1170
2	0.0000	0.0000	0.0000	0.0000	0.0000	0.0031	0.0000	0.0031	0.0031*
3	0.0630	0.0418	0.0513	0.1562	0.0000	0.0000	0.0000	0.0000	0.1562
4	0.0130	0.0238	0.0276	0.0645	0.0000	0.0015	0.0000	0.0015	0.0661
5	0.0477	0.0299	0.0394	0.1170	0.0000	0.0000	0.0000	0.0000	0.1170
6	0.0130	0.0207	0.0245	0.0582	0.0000	0.0015	0.0000	0.0015	0.0598
7	0.0630	0.0418	0.0513	0.1562	0.0000	0.0000	0.0000	0.0000	0.1562
8	0.0000	0.0031	0.0031	0.0062	0.0000	0.0031	0.0000	0.0031	0.0094
9	0.0399	0.0373	0.0276	0.1049	0.0000	0.0031	0.0000	0.0031	0.1080
10	0.0130	0.0238	0.0276	0.0645	0.0000	0.0015	0.0000	0.0015	0.0661
11	0.0399	0.0242	0.0164	0.0806	0.0000	0.0015	0.0000	0.0015	0.0821
12	0.0000	0.0000	0.0000	0.0000	0.0000	0.0031	0.0000	0.0031	0.0031*
13	0.0399	0.0373	0.0276	0.1049	0.0000	0.0031	0.0000	0.0031	0.1080
14	0.0130	0.0207	0.0245	0.0582	0.0000	0.0015	0.0000	0.0015	0.0598
15	0.0399	0.0242	0.0164	0.0806	0.0000	0.0015	0.0000	0.0015	0.0821

Tabelle 2: Ähnlichkeitsbewertung der Differenzbilder

struktur ist zwar zumindest für den etwas geübten Betrachter zu erkennen, allerdings sind eine Vielzahl zusätzlicher und störender Elemente enthalten.

Um dieses Problem zu beseitigen, gibt es in der aktuellen Softwarelösung die Möglichkeit, im Voxelmodell Bereiche zu markieren und die entsprechenden Voxel zu entfernen. Die Abb. 42 zeigt das Ergebnis für das hier verwendete Modell. An dieser Stelle bieten sich natürlich weitere Verbesserungen an, indem dem Benutzer ein möglichst komfortables Werkzeug bereitgestellt wird, um diese Bereinigung vorzunehmen. Da es sich bei den zu entfernenden Strukturen im wesentlichen um die gleichen wiederkehrenden Konstellationen handelt, wie sie auch während des Fleshing-Out-Projections-Algorithmus auftreten, ist es sicherlich auch möglich, hier weitgehend automatisch vorzugehen oder zumindest dem Benutzer zu ermöglichen, durch einen einzelnen Klick eine solche Struktur zu markieren und sie dann automatisch zu entfernen.

Auf diese bereinigte Struktur können nun die Techniken zum Finden von Features aus den Abschnitten 5.2 und 5.3 angewendet werden. Das automatische Finden von Quadern liefert hier bei geeigneter Parameterwahl insgesamt 139 mögliche Quader. In der vorliegenden Implementierung kann sich der Benutzer aus dieser recht großen Menge die für die weitere Verarbeitung geeigneten auswählen. Momentan geschieht das durch Durchblättern aller

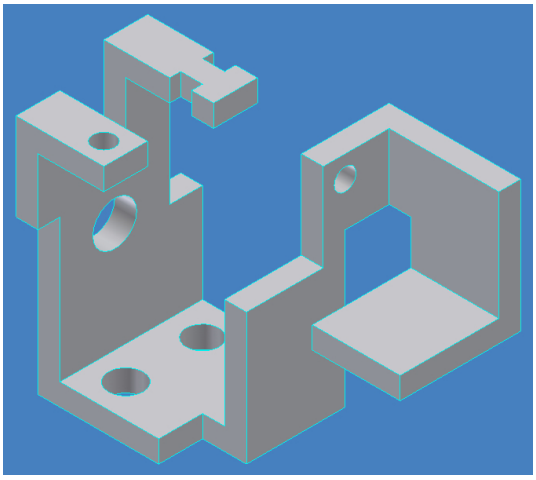


Abbildung 39: Ein CAD-Teil.

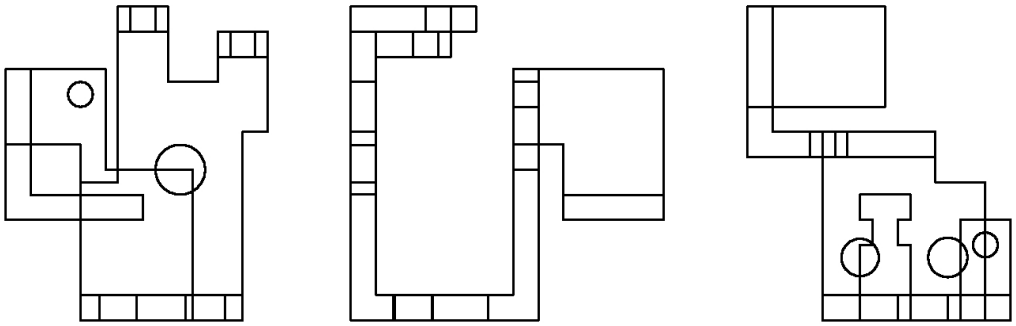


Abbildung 40: Die Projektionen des Teils aus Abb. 39.

gefundenen Lösungen und Löschen der nicht benötigten. Hier sind ebenfalls weitere Verbesserungen in der Bedienoberfläche denkbar und auch notwendig. Möglich wäre z. B., beim Bewegen der Maus über das Modell immer genau die Quader anzuzeigen, in deren Bild sich die aktuelle Mausposition befindet. Auf diese Weise könnte der Benutzer die geometrische Struktur des Teils schrittweise aufarbeiten und bekommt jeweils nur eine sehr kleine Anzahl von Quadern präsentiert, zwischen denen er sich entscheiden kann und muss.

Hier wurden aus den gefundenen 139 Quadern 14 ausgewählt, sie sind in den Abbn. 43 und 44 gezeigt. Da die Ergebnisse der automatischen Suche nicht die gesamte Geometrie des Teils abdecken, wurden noch zwei Qua-

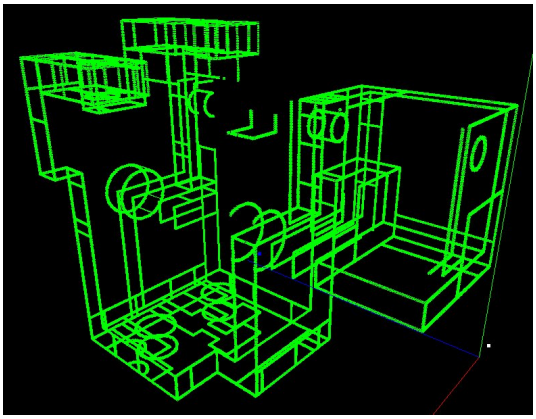


Abbildung 41: Das aus den Projektionen gebildete Voxelmodell.

der manuell mittels der Akkumulationstechnik ausgewählt (Abb. 45). Nach der Identifikation der Quader kann das automatische Finden von Zylindern angewendet werden, wobei hier fünf Objekte gefunden werden (Abb. 46).

Diese Elemente können nun als Primitive für die Teilegenerierung weiterverwendet werden. Allerdings besteht an dieser Stelle das Problem, dass durch auftretende Ungenauigkeiten beim Finden der Objekte Werte für die Koordinaten, die eigentlich gleich sein sollten, nicht übereinstimmen. So eine Situation tritt z. B. bei benachbarten Quadern auf, die koplanare Seitenflächen besitzen sollten. Diese Abweichungen würden später bei der Modellbildung und Ableitung der Projektionen zu Fehlern führen. Um das Problem zu beheben, wird separat für jede Koordinatenrichtung ein Clustering der Koordinatenwerte durchgeführt. Für die praktische Anwendung stellt diese geringfügige Veränderung der Werte kein Problem dar, da aus einer gescannten Zeichnung ohnehin kein exaktes Modell gewonnen werden kann, sondern dazu eine Auswertung und Berücksichtigung der vorhandenen Bemaßungen notwendig wäre.

Nach diesem Zwischenschritt kann nun das Erzeugen der Teilekandidaten und die Bewertung erfolgen. An dieser Stelle zeigt sich auch die Wirksamkeit der in Abschnitt 5.4 beschriebenen Optimierungsmöglichkeiten für die Teilesuche sehr deutlich. Während die alleinige Verwendung der Prozedur `GenerateParts()` ohne Modifikationen und Vorverarbeitung hier bereits nur für die 15 gefundenen Quader über 60000 verschiedene Lösungen liefern würde (aufgrund der praktischen Unbrauchbarkeit wurde die Berechnung ab-

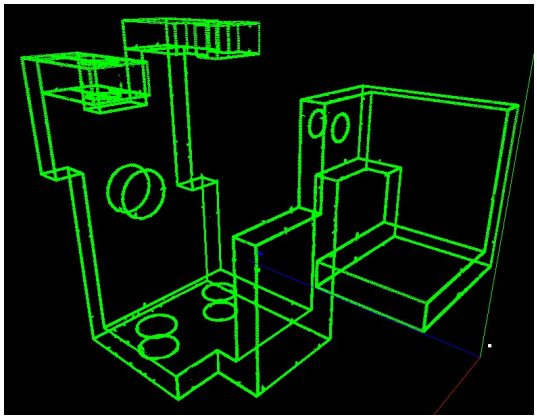


Abbildung 42: Das bereinigte Voxelmodell.

gebrochen, die genaue Zahl dürfte weit höher liegen), kann diese Zahl durch die Optimierungen auf 96 reduziert werden. Der in der Vorverarbeitung entstehende Graph mit den identifizierten Brückenelementen ist in Abb. 47 zu sehen. Die erhaltenen 96 Varianten können dann zur Teilebewertung weiterverwendet werden, um letztendlich das gesuchte Teil erfolgreich zu identifizieren.

Aufgrund der Implementierung als Prototyp und der Verwendung mehrerer voneinander getrennter Softwarekomponenten sowie technischer Probleme mit der Software OpenCASCADE kann die Laufzeit des Verfahrens schwer allgemein beurteilt werden. Sie ist auch sehr stark von den vom Benutzer gemachten Auswahlen abhängig. Der mit weitem Abstand zeitaufwändigste Teil ist dabei im Moment die Bewertung der Teilekandidaten, sie benötigt allein schon für das oben angegebene Beispiel mit 16 Kandidaten mehrere Minuten, während die übrigen Schritte wie das Finden der Elemente und die Generierung der Teilekandidaten innerhalb weniger Sekunden abgeschlossen sind. Hier besteht allerdings durch den Einsatz einer anderen Software für die Geometrieerzeugung und Projektionsableitung ein erhebliches Verbesserungspotential, so dass letztendlich für den Benutzer ein angenehmes Arbeiten möglich wird.

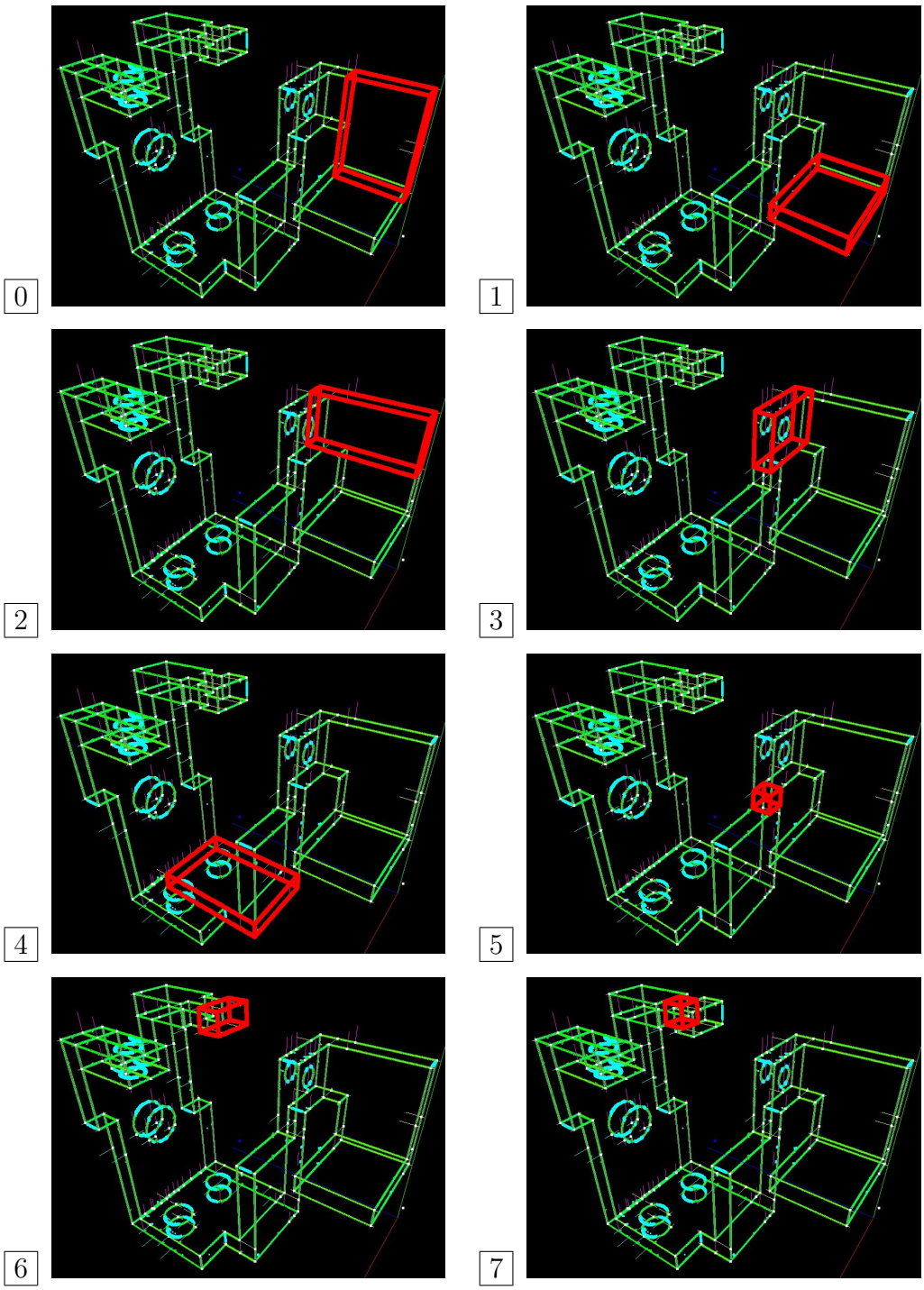


Abbildung 43: Durch das automatische Verfahren gefundene Quader (1).

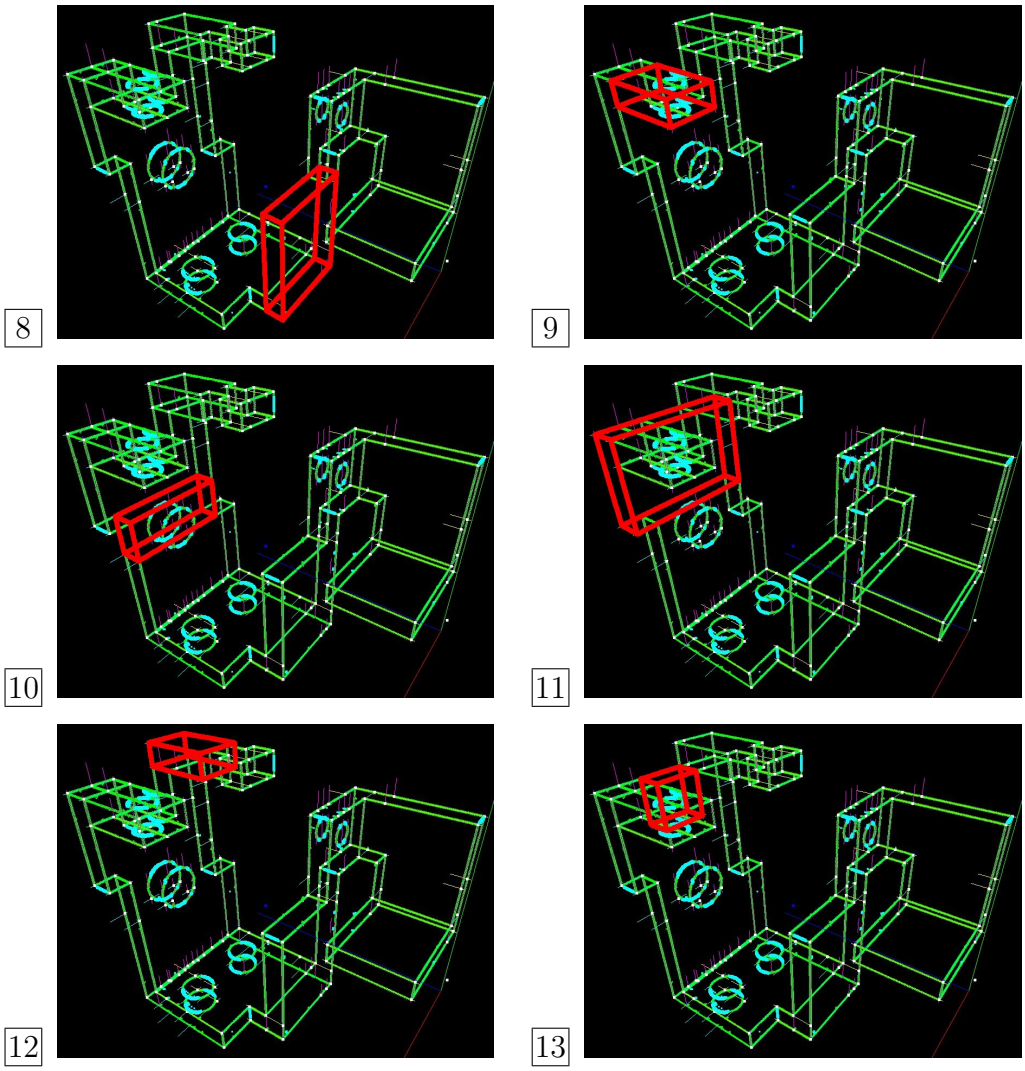


Abbildung 44: Durch das automatische Verfahren gefundene Quader (2).

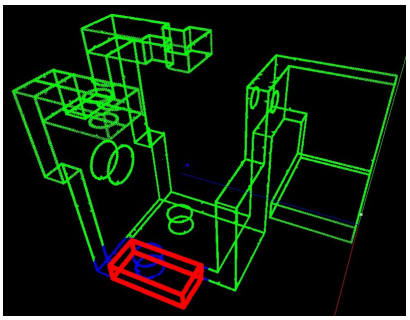
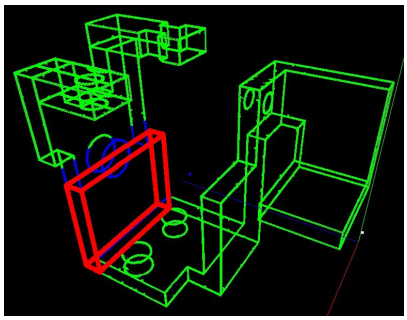
14  15 

Abbildung 45: Manuell gefundene Quader.

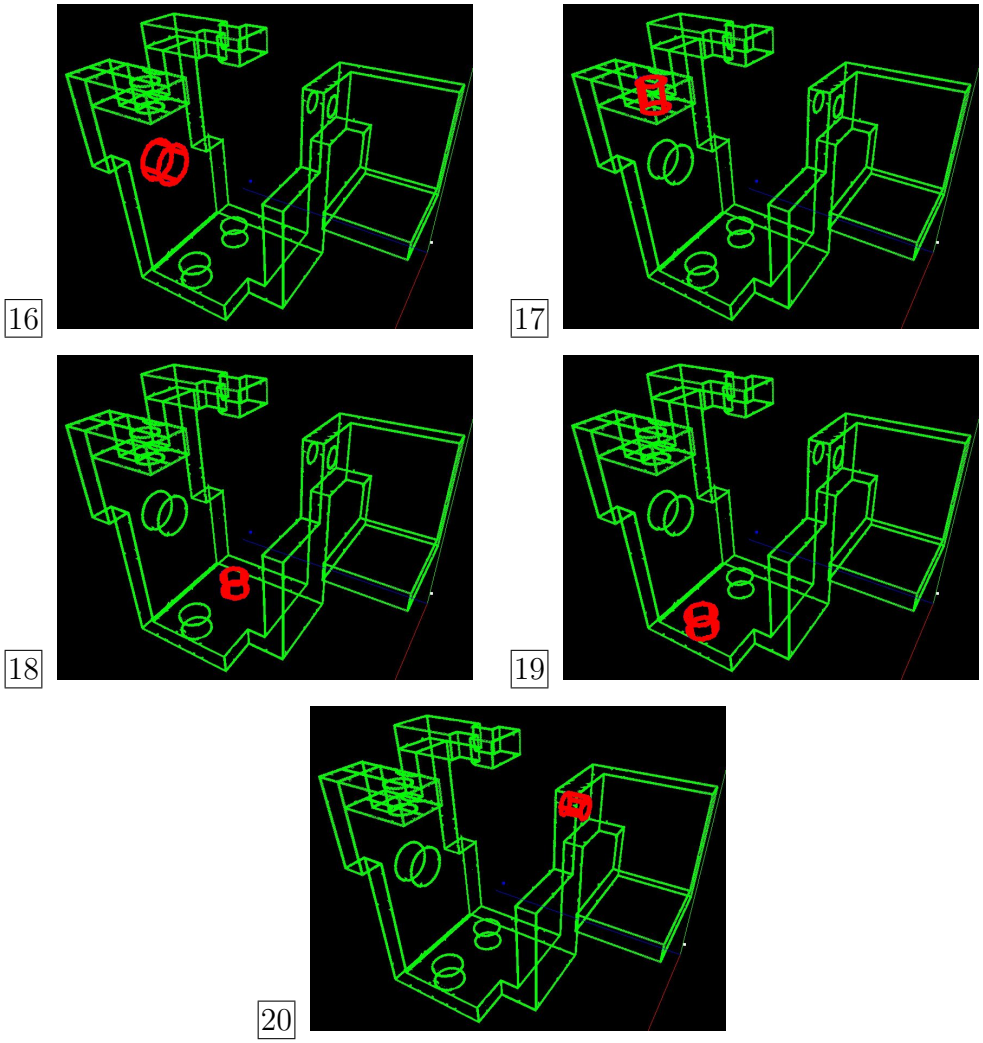


Abbildung 46: Gefundene Zylinder.

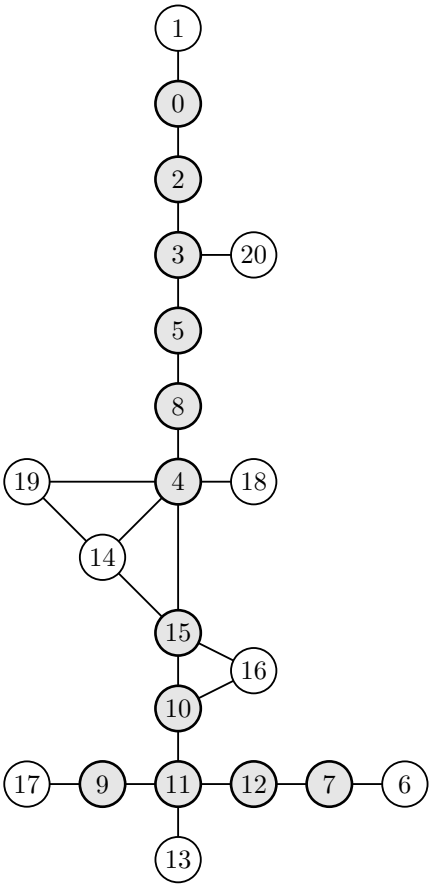


Abbildung 47: Graph für die gefundenen Primitive mit markierten Brückenelementen.

6 Zusammenfassung

In diesem ersten Teil der Arbeit wurde eine neuartige Lösung für die Rekonstruktion von 3D-CAD-Modellen aus gescannten 2D-Papierzeichnungen vorgeschlagen. Die grundlegende Idee besteht dabei darin, unter Umgehung einer Vektorisierung direkt aus den gescannten Projektionen ein Voxelmodell zu erzeugen, auf das dann verschiedene Methoden zum Finden von Features angewendet werden können. Aus den gefundenen Features lassen sich dann Teilekandidaten kombinieren, deren 2D-Projektionen mit den Originalprojektionen abgeglichen werden, um das Lösungsteil zu finden. Die Arbeitsschritte des Verfahrens sind hier noch einmal aufgeführt:

1. Scannen,
2. Entfernen von Elementen, die nicht zu Projektionen gehören,
3. Ausrichten der Projektionen,
4. Voxelmodell bilden,
5. Histogramm des Voxelmodells auswerten / Bereich auswählen,
6. Features finden und auswählen,
7. Features zu Teilekandidaten kombinieren,
8. Projektionen erzeugen und mit Originalen abgleichen,
9. Kandidaten bewerten und Lösung auswählen.

Ziel dieses Teils der Arbeit war es, den prinzipiellen Lösungsweg aufzuzeigen (veröffentlicht in [6]). Selbstverständlich bietet jedes einzelne Modul zahlreiche Möglichkeiten für Verbesserungen und Optimierungen. Dabei kommt es vor allem auch darauf an, den im Moment noch sehr hohen Anteil an Arbeit für den Benutzer möglichst weit zu verringern und ihm eine komfortable und attraktive Benutzeroberfläche zur Verfügung zu stellen, um die Lösung für den produktiven Einsatz tauglich zu machen. Dies wird im Rahmen einer weiteren Produktentwicklung, für die diese Arbeit die theoretische Grundlage bildet, auch geschehen.

Teil II

Fitting dreidimensionaler Formen

7 Fitting durch Normalisierung

In diesem zweiten Teil der Arbeit werden Methoden zum Fitten dreidimensionaler Objekte vorgestellt, die das Prinzip der Normalisierung verwenden. Die Objekte sind dabei als Menge von Voxeln gegeben, die deren Inneres ausfüllen. Die Voxel sollen hier alle die gleiche konstante Intensität 1 besitzen.

Es gibt zahlreiche Anwendungen, die 3D-Voxeldaten liefern, z. B. Verfahren wie die Computertomographie, und bei denen die Aufgabe besteht, Objekte in diesen Daten zu finden oder ein Modell möglichst gut an eine Datenmenge anzupassen.

Der nächste Abschnitt erläutert zunächst kurz das Grundprinzip des Fittings mittels Normalisierung (s. a. [22, 23]), in den weiteren Abschnitten wird dieses Prinzip für die dritte Dimension erweitert und es werden Fittingalgorithmen für verschiedene dreidimensionale Formen angegeben (veröffentlicht in [4, 5]).

7.1 Grundprinzip

Gegeben sei eine Klasse $\mathbf{T}$ von Transformationen, wobei jede Transformation $t \in \mathbf{T}$ durch n Parameter $\xi_1, \ldots, \xi_n$ beschrieben wird. Gegeben sei weiterhin eine Klasse von Primitiven, die die theoretische Objektform beschreiben, die angepasst werden soll (Quader, Kugel, Ellipsoid etc.). Ein Primitiv $P(\theta)$ wird durch m Parameter $\theta_1, \ldots, \theta_m$ beschrieben.

Wenn ein Objekt O gegeben ist, so bestimmt man Merkmale

$$\mathbf{f}(O) = (f_1, f_2, f_3, \ldots).$$

Ein Fitting des Objekts kann nun erfolgen, indem man die Optimierungsaufgabe

$$||\mathbf{f}(O) - \mathbf{f}(P(\theta))||^2 \longrightarrow \text{minimum}$$

löst, wobei der m-dimensionale Raum Θ aller Primitive zu durchsuchen ist.

Für das Fitting durch Normalisierung ist zunächst eine zu den Klassen der Primitive und Transformationen passende Standardlage zu wählen, z. B. ein Einheitsquadrat für die Klasse aller Quadrate und die Klasse der Ähnlichkeitstransformationen. Danach wird das Objekt durch Bestimmung einer geeigneten Transformation auf diese Standardlage normalisiert. Sei $P(\theta^*)$ das optimale Primitiv für das Objekt O, so muss (da beide Objekte der gleichen Transformation unterzogen werden) die Optimierungsaufgabe

$$||\mathbf{f}(O') - \mathbf{f}(P'(\theta^*))||^2 \longrightarrow \text{minimum}$$

gelöst werden. Sie hat lediglich die Dimension $m - n$. In [22] bzw. [23] sind Beispiele mit Dimension 0 bzw. 1 für das Fitting zweidimensionaler Objekte angegeben.

Das optimale Primitiv $P(\theta^*)$ erhält man letztendlich durch Rücktransformation des Lösungsprimitivs der Optimierungsaufgabe.

7.2 Transformationsbestimmung

Eine zentrale Aufgabe bei der Normalisierung ist natürlich die Berechnung der Transformation, die das Objekt auf die gewünschte Standardlage abbildet. In diesem Abschnitt soll die prinzipielle Vorgehensweise am Beispiel der affinen Transformationen in der Ebene gezeigt werden. Dabei wird eine Separation der affinen Abbildung genutzt, und durch passend vorgegebene Normalisierungsbedingungen werden die einzelnen Parameter der Abbildung schrittweise ermittelt (siehe [21]). An dieser Stelle wird dabei davon ausgegangen, dass das Objekt bereits auf die Standardlage zentriert wurde, so dass die zu betrachtende affine Abbildung keinen Translationsanteil mehr aufweist.

Für die Normalisierung nutzen wir die Flächenmomente

$$m_{pq} = \iint\limits_{object} x^p y^q dxdy \quad (p, q \in \mathbb{N}),$$

bzw. später für die Implementierung ihre diskreten Äquivalente

$$M_{pq} = \sum_{(x,y)} \sum_{\in object} x^p y^q.$$

Wir verwenden hier die Separation einer affinen Abbildung $\mathbf{A}$ in eine Scherung $\mathbf{D}$, eine anisotrope Skalierung $\mathbf{S}$ sowie eine Rotation $\mathbf{R}$:

$$\begin{aligned}
\mathbf{A} &= \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \\
&= \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} \beta & 0 \\ 0 & \gamma \end{pmatrix} \begin{pmatrix} 1 & \alpha \\ 0 & 1 \end{pmatrix} \\
&= \mathbf{R} \cdot \mathbf{S} \cdot \mathbf{D}
\end{aligned}$$

Es gilt also:

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} = \begin{pmatrix} \beta\cos\varphi & \alpha\beta\cos\varphi - \gamma\sin\varphi \\ \beta\sin\varphi & \alpha\beta\sin\varphi + \gamma\cos\varphi \end{pmatrix}.$$

Um zu zeigen, dass diese Separation möglich ist, können die Parameter der einzelnen Transformationen aus den Werten a_{11}, a_{12}, a_{21}, a_{22} bestimmt werden. Zunächst gilt

$$\beta = \sqrt{a_{11}^2 + a_{21}^2}$$

(wir gehen hier von $\beta > 0$ aus), und damit kann dann auch φ aus

$$\sin\varphi = \frac{a_{21}}{\sqrt{a_{11}^2 + a_{21}^2}}, \quad \cos\varphi = \frac{a_{11}}{\sqrt{a_{11}^2 + a_{21}^2}}$$

berechnet werden. Die beiden übrigen Parameter α und γ erhält man unter Verwendung von a_{12} und a_{22} dann gemäß

$$\alpha = \frac{a_{11}a_{12} + a_{21}a_{22}}{a_{11}^2 + a_{21}^2}$$

$$\gamma = \frac{a_{11}a_{22} - a_{12}a_{21}}{\sqrt{a_{11}^2 + a_{21}^2}}.$$

Es gilt hierbei, dass sich immer eine Lösung mit $\beta > 0$ und $\gamma > 0$ finden lässt, wenn die Determinante $\det(\mathbf{A}) > 0$ ist. Ist $\det(\mathbf{A}) < 0$, so kann

durch zusätzliche Berücksichtigung einer Spiegelung ebenfalls eine Zerlegung gefunden werden, die eine „positive“ anisotrope Skalierung enthält:

$$\mathbf{A} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \mathbf{R} \cdot \mathbf{S} \cdot \mathbf{D}.$$

Diese Separation ist zusammen mit der nachfolgend beschriebenen Bestimmung der Transformationsparameter anhand von Bedingungen an die Momente auch als sogenannte „Standardmethode“ bekannt [25]. Um die Parameter zu erhalten betrachten wir jeweils, wie sich die Momente bei der im aktuellen Schritt auszuführenden Transformation verändern und versuchen, durch Vorgabe geeigneter Normalisierungsbedingungen eine Berechnung möglich zu machen.

Im ersten Schritt, der X-Scherung $\mathbf{D}$, transformieren sich die Momente wie folgt:

$$\begin{aligned} m'_{pq} &= \iint\limits_{object} (x + \alpha y)^p y^q dx dy \\ &= \sum_{k=0}^{p} \binom{p}{k} \alpha^{p-k} m_{k,p+q-k}. \end{aligned}$$

Für die Berechnung des Parameters α wird das Moment

$$m'_{11} = m_{11} + \alpha m_{02} = 0$$

genutzt, und durch diese Bedingung erhält man

$$\alpha = -\frac{m_{11}}{m_{02}}.$$

Die X-Scherung mit dem soeben erhaltenen Parameter muss nun natürlich auch auf alle weiteren Momente angewendet werden, das heisst

$$\begin{aligned} m^{[D]}_{pq} &= \iint\limits_{object} \left(x - \frac{m_{11}}{m_{02}} y \right)^p y^q dx dy \\ &= \sum_{k=0}^{p} \binom{p}{k} \left(-\frac{m_{11}}{m_{02}} \right)^{p-k} m_{k,p+q-k}. \end{aligned}$$

Der nächste Schritt ist die anisotrope Skalierung **S**. Dabei transformieren sich die Momente wie folgt:

$$m'_{pq} = (\beta\gamma) \iint\limits_{object} (\beta x)^p (\gamma y)^q dxdy = \beta^{p+1}\gamma^{q+1} m_{pq}.$$

Für die Normierung werden die Bedingungen

$$\begin{aligned} m'_{20} &= \beta^3\gamma m_{20} = 1, \\ m'_{02} &= \beta\gamma^3 m_{02} = 1. \end{aligned}$$

verwendet. Mit $m_{20} \neq 0$ und $m_{02} \neq 0$ erhält man die Transformationsparameter

$$\beta = \sqrt[8]{\frac{m_{02}}{m_{20}^3}}, \quad \gamma = \sqrt[8]{\frac{m_{20}}{m_{02}^3}}.$$

Die negativen Werte der Wurzeln sind dabei auch mögliche Lösungen. Um eine Eindeutigkeit herbeizuführen, werden Spiegelungen an dieser Stelle ausgeschlossen.

Die weiteren Momente transformieren sich dann gemäß

$$m_{pq}^{[SD]} = \sqrt[8]{\left(m_{02}^{[D]}\right)^{p-3q-2} \left(m_{20}^{[D]}\right)^{q-3p-2}} \cdot m_{pq}^{[D]}.$$

Offensichtlich verletzt diese Transformation der Momente nicht die bisher erzielte Normierung $m_{11} = 0$.

Als letzter Schritt ist die Rotation auszuführen. Für eine Rotation um den Winkel φ transformieren sich die Momente gemäß

$$\begin{aligned} m'_{pq} &= \iint\limits_{object} (x\cos\varphi - y\sin\varphi)^p (x\sin\varphi + y\cos\varphi)^q dxdy \\ &= \sum_{k=0}^{p}\sum_{l=0}^{q} (-1)^k \binom{p}{k}\binom{q}{l} \cos^{p+1-k}\varphi \sin^{q+k-l}\varphi m_{p+q-k-l,k+l} \end{aligned}$$

In [25] wird zur Bestimmung des Rotationsparameters eine Summe von Momenten dritter Ordnung normiert. Die Momente dritter Ordnung transformieren sich wie folgt:

$$\begin{aligned}
m'_{30} &= m_{30}\cos^3\varphi - 3m_{21}\cos^2\varphi\sin\varphi + \ldots \\
&\quad \ldots + 3m_{12}\cos\varphi\sin^2\varphi - m_{03}\sin^3\varphi, \\
m'_{21} &= m_{21}\cos^3\varphi + (m_{30} - 2m_{12})\cos^2\varphi\sin\varphi + \ldots \\
&\quad \ldots + (m_{03} - 2m_{21})\cos\varphi\sin^2\varphi + m_{12}\sin^3\varphi, \\
m'_{12} &= m_{12}\cos^3\varphi + (-m_{03} + 2m_{21})\cos^2\varphi\sin\varphi + \ldots \\
&\quad \ldots + (m_{30} - 2m_{12})\cos\varphi\sin^2\varphi - m_{21}\sin^3\varphi, \\
m'_{03} &= m_{03}\cos^3\varphi + 3m_{12}\cos^2\varphi\sin\varphi + \ldots \\
&\quad \ldots + 3m_{21}\cos\varphi\sin^2\varphi + m_{30}\sin^3\varphi.
\end{aligned}$$

Für die Normierung kann die Summe

$$m'_{30} + m'_{12} = (m_{30} + m_{12})\cos\varphi - (m_{03} + m_{21})\sin\varphi = 0$$

benutzt werden, womit man

$$\tan\varphi = \frac{\sin\varphi}{\cos\varphi} = \frac{m_{30} + m_{12}}{m_{03} + m_{21}}$$

und damit zwei mögliche, durch Drehung um 180° voneinander verschiedene Lösungen erhält, z. B.

$$\begin{aligned}
\sin\varphi &= \frac{m_{30} + m_{12}}{\sqrt{(m_{30} + m_{12})^2 + (m_{03} + m_{21})^2}} \\
\cos\varphi &= \frac{m_{03} + m_{21}}{\sqrt{(m_{30} + m_{12})^2 + (m_{03} + m_{21})^2}}
\end{aligned}$$

Eine explizite Berechnung des Winkels ist dabei gar nicht notwendig, die Terme für $\sin\varphi$ und $\cos\varphi$ können direkt in die Transformationsgleichungen eingesetzt werden:

$$\begin{aligned} m_{pq}^{[RSD]} &= \left((m_{30}^{[SD]} + m_{12}^{[SD]})^2 + (m_{03}^{[SD]} + m_{21}^{[SD]})^2\right)^{-\frac{p+q}{2}} \cdot \ldots \\ &\quad \ldots \cdot \sum_{k=0}^{p} \sum_{l=0}^{q} \left((-1)^k \binom{p}{k} \binom{q}{l} (m_{03}^{[SD]} + m_{21}^{[SD]})^{p+l-k} \cdot \ldots \right. \\ &\quad \left. \ldots \cdot (m_{30}^{[SD]} + m_{21}^{[SD]})^{q+k-l} m_{p+q-k-l,k+l}^{[SD]} \right). \end{aligned}$$

Durch die X-Scherung und die anisotrope Skalierung wurden die Normierungsbedingungen

$$m_{20} = 1, \quad m_{11} = 0, \quad m_{02} = 1$$

erfüllt. Diese Momente bestimmen aber die Trägheitsellipse des Objekts, die durch diese Normierung zu einem Kreis wird. Damit ist sichergestellt, dass eine nachfolgende Rotation diese vorher erzielte Normierung nicht wieder zerstört.

Die Trägheitsellipse ist ein wesentliches Merkmal eines Objekts. Sie kann z. B. für die Bestimmung einer definierten Orientierung eines Objekts eingesetzt werden, um etwa einen Lagevergleich für zwei Objekte vorzunehmen. In enger Verbindung zur Trägheitsellipse steht dabei die Funktion

$$m_{20}(\varphi) = m_{20}\cos^2\varphi + 2m_{11}\cos\varphi\sin\varphi + m_{02}\sin^2\varphi. \tag{1}$$

Um eine Orientierung für das Objekt zu definieren, kann z. B. das Minimum dieser Funktion genutzt werden. Durch die Substitution

$$\sin\varphi = -\frac{x}{\sqrt{m_{20}(\varphi)}}, \qquad \cos\varphi = \frac{y}{\sqrt{m_{20}(\varphi)}}$$

erhält man daraus die Gleichung der Trägheitsellipse:

$$1 = m_{20}y^2 - 2m_{11}xy + m_{02}x^2.$$

Die Trägheitsellipse besitzt jedoch den Nachteil, auch für manche Objekte, die keinerlei Symmetrie aufweisen, zu einem Kreis zu entarten, so dass

sie in diesem Fall für eine Orientierungsbestimmung nicht mehr nutzbar ist. Dass es solche unsymmetrischen Objekte mit dieser Eigenschaft geben muss, wird schnell klar, wenn man ein beliebiges Objekt der oben angegebenen Normalisierung unterwirft. In diesem Fall kann mit einer Verallgemeinerung von (1) gearbeitet werden:

$$m_{n,0}(\varphi) = \sum_{k=0}^{n} \binom{n}{k} m_{n-k,k} \cos^{n-k}\varphi \sin^{k}\varphi, \qquad n = 2, 3, 4, \ldots.$$

Für $n = 3$ und $n = 4$ nimmt diese Funktion folgende Gestalt an:

$$\begin{aligned} m_{3,0}(\varphi) &= m_{30}\cos^3\varphi + 3m_{21}\cos^2\varphi\sin\varphi + \ldots \\ &\quad \ldots + 3m_{12}\cos\varphi\sin^2\varphi + m_{30}\sin^3\varphi, \\ m_{4,0}(\varphi) &= m_{40}\cos^4\varphi + 4m_{31}\cos^3\varphi\sin\varphi + \ldots \\ &\quad \ldots + 6m_{22}\cos^2\varphi\sin^2\varphi + 4m_{13}\cos\varphi\sin^3\varphi + \ldots \\ &\quad \ldots + m_{04}\sin^4\varphi. \end{aligned}$$

In [19] werden diese Funktionen näher untersucht, und es wird gezeigt, wie sie ergänzt durch geeignete Entscheidungskriterien erfolgreich zur Bestimmung der Orientierung von Objekten mit degenerierter Trägheitsellipse eingesetzt werden können.

8 Fitting von Quadern

Nach diesen kurzen Ausführungen zu den Grundlagen und einem Beispiel für eine Separation soll nun der erste Algorithmus vorgestellt werden. In [24] wurde bereits ein Verfahren für das Fitting von Rechtecken bzw. Quadraten in 2D angegeben. Dieses wird hier auf die dritte Dimension erweitert, was aufgrund der hinzukommenden Mehrdeutigkeiten bei der Rotationsnormalisierung mit einigen zusätzlichen Schwierigkeiten verbunden ist.

Quader können durch neun Parameter beschrieben werden: Drei Parameter für die Seitenlängen, drei für die Lage des Mittelpunkts sowie drei für die Rotation. Als Transformationen verwenden wir hier die Gruppe der Ähnlichkeitstransformationen. Eine solche Ähnlichkeitstransformation kann durch sieben Parameter beschrieben werden: Drei für den Translationsanteil, drei für den Rotationsanteil und einer für die isotrope Skalierung.

Als Standardlage für Quader verwenden wir einen achsenparallelen, um den Koordinatenursprung zentrierten Quader mit dem Volumen 1. Er kann durch zwei Parameter, z. B. zwei Seitenlängen, beschrieben werden. Dies ist dann auch die Dimension des zu lösenden Optimierungsproblems.

Für das Fitting benutzen wir als Merkmale die Volumenmomente

$$m_{pqr} = \iiint\limits_{object} x^p y^q z^r dx dy dz$$

bis vierter Ordnung ($p + q + r \leq 4$).

8.1 Normalisierung

Die Normalisierung verläuft in folgenden Schritten:

8.1.1 Translation

Zunächst normalisieren wir die Position des Objekts durch eine Translation T:

$$(t_x, t_y, t_z) = \left(-\frac{m_{100}}{m_{000}}, -\frac{m_{010}}{m_{000}}, -\frac{m_{001}}{m_{000}} \right).$$

Ab sofort verwenden wir die zentralen Momente $m'_{pqr} = \mu_{pqr}$ und es gilt

$$m'_{100} = m'_{010} = m'_{001} = 0.$$

8.1.2 Rotation

Um die Lage des Objekts zu normalisieren, wird es so gedreht, dass die Achsen des Trägheitsellipsoids mit den Koordinatenachsen übereinstimmen. Diese Achsen sind die Eigenvektoren der Trägheitsmatrix

$$I = \begin{pmatrix} m'_{200} & m'_{110} & m'_{101} \\ m'_{110} & m'_{020} & m'_{011} \\ m'_{101} & m'_{011} & m'_{002} \end{pmatrix}.$$

Sie ist reell und symmetrisch und besitzt drei reelle Eigenwerte und drei paarweise senkrechte Eigenvektoren. Hier können Mehrdeutigkeiten entstehen, da mit v auch $-v$ ein Eigenvektor der Matrix ist, und es gibt demzufolge acht mögliche Rotationen. In [9] wird eine Heuristik angegeben, um eine davon auszuwählen. Dies ist notwendig, wenn z. B. die Form von Objekten verglichen werden soll, aber da hier ausschließlich Quader betrachtet werden, ist die Auswahl der Rotation hier nicht von Bedeutung. Die Matrix für die Rotation wird gebildet, indem die ausgewählten Eigenvektoren als Zeilen eingesetzt werden.

Für die Rotation R mit

$$R = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix}$$

(und $\det(R) = 1$) werden die Momente unter Verwendung von

$$\begin{aligned} (x+y+z)^n &= \sum_{\substack{0 \le a,b,c \le n \\ a+b+c=n}} \frac{(a+b+c)!}{a!\,b!\,c!} x^a y^b z^c \\ &= \sum_{\substack{0 \le a,b,c \le n \\ a+b+c=n}} \binom{a+b+c}{b+c} \binom{b+c}{c} x^a y^b z^c \end{aligned}$$

(siehe [11]) wie folgt transformiert:

$$\begin{aligned} m''_{pqr} \;=\; & \sum_{\substack{0\le a,b,c\le p\\ a+b+c=p}} \sum_{\substack{0\le d,e,f\le q\\ d+e+f=q}} \sum_{\substack{0\le h,i,k\le r\\ h+i+k=r}} \binom{a+b+c}{b+c} \cdot \ldots \\ & \ldots \cdot \binom{b+c}{c}\binom{d+e+f}{e+f}\binom{e+f}{f}\binom{h+i+k}{i+k} \cdot \ldots \\ & \ldots \cdot \binom{i+k}{k} \cdot R^a_{11}R^b_{12}R^c_{13}R^d_{21}R^e_{22}R^f_{23}R^h_{31}R^i_{32}R^k_{33} \cdot \ldots \\ & \ldots \cdot m'_{a+d+h,b+e+i,c+f+k}. \end{aligned}$$

Danach gilt $m''_{110} = m''_{101} = m''_{011} = 0$.

8.1.3 Skalierung

Anschließend wird die Größe des Objekts normalisiert, so dass danach $m'''_{000} = 1$ gilt. Der dazu notwendige Skalierungsfaktor ist

$$\alpha = \sqrt[3]{\frac{1}{m''_{000}}}.$$

Die Momente werden mittels der Beziehung

$$m'''_{pqr} = \alpha^{p+q+r+3} m''_{pqr}$$

transformiert.

8.1.4 Standardlagenparameter bestimmen

Um die beiden freien Parameter zu bestimmen, werden einige höhere Momente mit nichttrivialen Werten benutzt. Wenn a, b und c die Seitenlängen eines Quaders in Standardlage bezeichnen ($abc = 1$), so können die folgenden Momente in Abhängigkeit von a und b angegeben werden (die Momente dritter Ordnung sind gleich 0):

$$\mu^q_{200} = \frac{a^2}{12} \qquad \mu^q_{400} = \frac{a^4}{80} \qquad \mu^q_{220} = \frac{a^2 b^2}{144}$$
$$\mu^q_{020} = \frac{b^2}{12} \qquad \mu^q_{040} = \frac{b^4}{80} \qquad \mu^q_{202} = \frac{1}{144 b^2}$$
$$\mu^q_{002} = \frac{1}{12 a^2 b^2} \qquad \mu^q_{004} = \frac{1}{80 a^4 b^4} \qquad \mu^q_{022} = \frac{1}{144 a^2}$$

Um a und b zu bestimmen, wird folgendes Optimierungsproblem gelöst:

$$\begin{aligned} f(a,b) \quad = \quad & (m'''_{200} - \mu^q_{200})^2 + (m'''_{020} - \mu^q_{020})^2 + (m'''_{002} - \mu^q_{002})^2 + \ldots \\ & \ldots + (m'''_{400} - \mu^q_{400})^2 + (m'''_{040} - \mu^q_{040})^2 + (m'''_{004} - \mu^q_{004})^2 + \ldots \\ & \ldots + (m'''_{220} - \mu^q_{220})^2 + (m'''_{202} - \mu^q_{202})^2 + (m'''_{022} - \mu^q_{022})^2 \\ \longrightarrow & \text{ minimum} \end{aligned}$$

$$\text{mit } 0 < a, b < \infty.$$

Damit können nun auf einfache Weise die Ecken des Quaders in Standardlage bestimmt werden, und man erhält den an die Voxelmenge optimal angepassten Quader durch Anwendung der inversen Transformation $(S \cdot R \cdot T)^{-1} = T^{-1} \cdot R^{-1} \cdot S^{-1}$.

Allerdings ist es nicht immer möglich, die Normalisierung auf die beschriebene Weise vorzunehmen. Das Trägheitsellipsoid kann zu einer Kugel oder zu einem Ellipsoid mit zwei Achsen gleicher Länge entarten. In beiden Fällen sind andere Methoden zur Normalisierung der Rotation nötig, die im folgenden beschrieben werden.

8.2 Spezialfälle

8.2.1 Kugel

Eine mögliche Entartung des Trägheitsellipsoids ist die Kugel. In diesem Fall werden Momente vierter Ordnung für die Rotationsnormalisierung verwendet. Um die korrekten Rotationsparameter zu finden benutzen wir die

experimentell ermittelte Eigenschaft der Momente vierter Ordnung, dass für einen im Ursprung zentrierten Würfel der Term $\mu_{400} + \mu_{040} + \mu_{004}$ genau dann minimal wird, wenn der Würfel achsenparallel positioniert ist (das ist auch die Position unserer gewählten Standardlage). Abb. 48 zeigt die Minima dieses Terms in Abhängigkeit von den drei Rotationswinkeln $r_x, r_y, r_z \in \mathbb{R}$, die für die Rotation $R_z \cdot R_y \cdot R_x$ benutzt wurden: Dies sind zum einen die Gitterpunkte $\{(\frac{\pi}{2}i, \frac{\pi}{2}j, \frac{\pi}{2}k) : i, j, k \in \mathbb{Z}\}$, zum anderen die beiden Geradenscharen $\{r_y = \frac{\pi}{2} + 2\pi i,\ r_z = -r_x + \frac{\pi}{2}j : i, j \in \mathbb{Z}\}$ und $\{r_y = \frac{3\pi}{2} + 2\pi i,\ r_z = r_x + \frac{\pi}{2}j : i, j \in \mathbb{Z}\}$. Um nun die Rotationsparameter für $R = R_z \cdot R_y \cdot R_x$ zu bestimmen, wird das Optimierungsproblem

$$\mu_{400}(r_x, r_y, r_z) + \mu_{040}(r_x, r_y, r_z) + \mu_{004}(r_x, r_y, r_z) \longrightarrow \text{minimum}$$

gelöst.

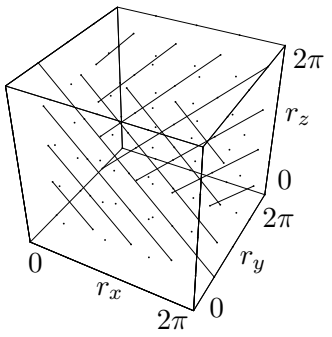


Abbildung 48: Minima von $\mu_{400}(r_x, r_y, r_z) + \mu_{040}(r_x, r_y, r_z) + \mu_{004}(r_x, r_y, r_z)$.

8.2.2 Rotationssymmetrie

Eine zweite Möglichkeit der Entartung besteht darin, dass zwei Achsen des Trägheitsellipsoids die gleiche Länge haben und die dritte Länge davon verschieden ist. In diesem Fall bringen wir die Achse mit der dritten, von den anderen beiden verschiedenen Länge mit der x-Achse zur Deckung. Um die korrekte Rotation um die x-Achse zu finden, verwenden wir wieder die Momente vierter Ordnung und führen die Optimierung

$$\mu_{040}(r_x) + \mu_{004}(r_x) \longrightarrow \text{minimum}$$

durch.

8.3 Ein Entscheidungskriterium

Bisher wurden drei verschiedene Fitting-Methoden vorgestellt. Um zu entscheiden, welche von ihnen jeweils angewendet wird, werden die Eigenwerte der Trägheitsmatrix herangezogen. Für diese Werte λ_1, λ_2 und λ_3 mit $\lambda_1 \geq \lambda_2 \geq \lambda_3$ betrachten wir die Verhältnisse $r_1 = |\frac{\lambda_2}{\lambda_1}|$ und $r_2 = |\frac{\lambda_3}{\lambda_2}|$. Sie sind invariant gegenüber Ähnlichkeitstransformationen und können benutzt werden, um zwischen den drei Fällen für die Rotationsnormalisierung zu unterscheiden:

- Sind beide Verhältnisse nahe 1, so ist das Objekt offensichtlich „würfelähnlich“ und für die Normalisierung sollten ausschließlich die Momente vierter Ordnung benutzt werden.
- Ist ein Verhältnis nahe 1 und das andere deutlich verschieden davon, sollte eine Richtung der Trägheitsmatrix entnommen werden und die beiden anderen sollten mittels der Momente vierter Ordnung bestimmt werden.
- In allen anderen Fällen ist die Trägheitsmatrix zu verwenden.

Abb. 49 veranschaulicht die Fitting-Qualität für jede der drei Methoden. Die Diagramme sind auf folgende Weise entstanden: In einem $200 \times 200 \times 200$-Voxel-Würfel werden zufällig Quader mit dem Volumen 50^3 erzeugt, d. h. die Voxel im Inneren des jeweiligen Quaders werden gesetzt, und anschließend wird mit jeder der drei Methoden ein Quader angepasst. Die gezeigte Fitting-Qualität q ist dabei das arithmetische Mittel der Abstände zwischen den Ecken des Original- und des gefitteten Quaders. Diese Zahl ist natürlich nur für kleine Werte (etwa kleiner als 10) aussagekräftig, größere Werte bedeuten ein völliges Versagen der jeweiligen Fitting-Methode. In der Abbildung sind in der oberen Reihe die Bereiche mit $0 \leq q \leq 1$ zu sehen, in der unteren Reihe die mit $1 < q$. Von links nach rechts wurden folgende Normalisierungsmethoden benutzt: Ausschließliche Verwendung der Trägheitsmatrix, ausschließliche Verwendung der vierten Momente, Entnahme einer Richtung aus der Trägheitsmatrix und Nutzung der Momente vierter Ordnung zur Bestimmung der beiden anderen Richtungen. Auf der x- bzw. y-Achse der Diagramme laufen die oben definierten Werte r_1 bzw. r_2 jeweils von 0 bis 1.

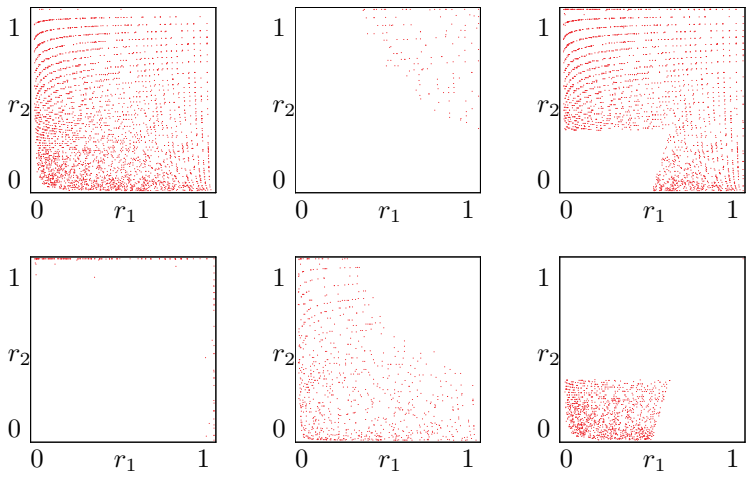


Abbildung 49: Fitting-Qualität für die drei verschiedenen Methoden zur Rotationsnormalisierung (Erläuterung siehe Text).

Die Diagramme lassen sich so interpretieren, dass in den Bereichen der oberen Reihe die jeweilige Fitting-Methode gut geeignet ist, in den Bereichen der unteren Reihe dagegen nicht eingesetzt werden sollte.

Die Entscheidung für eine der drei Methoden wird nun anhand des Schemas in Abb. 50 getroffen. Zunächst wird überprüft, ob beide Verhältnisse im Bereich I liegen. Ist dem so, werden die Momente vierter Ordnung zur Rotationsnormalisierung verwendet. Ist das nicht der Fall, wird geprüft, ob die Verhältnisse in einer der mit II markierten Regionen, und wenn ja, wird eine Richtung aus der Trägheitsmatrix entnommen und die beiden anderen werden mittels der Momente vierter Ordnung bestimmt. Andernfalls wird ausschließlich die Trägheitsmatrix zur Normalisierung verwendet.

Bei Experimenten haben sich $\varepsilon_1 = 0.5$ und $\varepsilon_2 = 0.75$ als brauchbare Werte erwiesen.

8.4 Experimentelle Ergebnisse

Es konnten sehr gute Fitting-Qualitäten erreicht werden, und das auch für den Fall, dass größere Regionen des Quaders fehlen. So kann zum Beispiel für Quader mit einem Volumen von 50^3, von denen eine in einer Ecke zentrierte

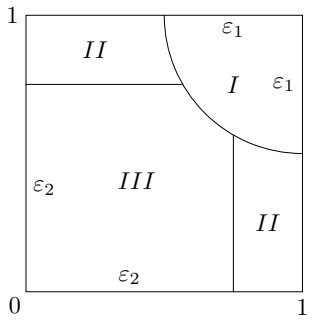


Abbildung 50: Die Regionen für die drei Methoden zur Rotationsnormalisierung.

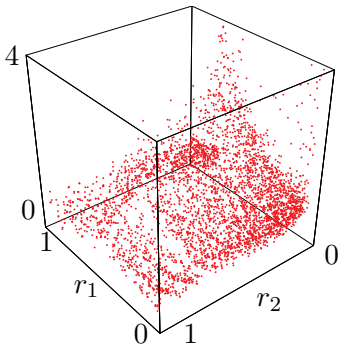


Abbildung 51: Fitting-Qualität für Quader mit einem Volumen von 50^3 und einer in einer Ecke zentrierten Kugel mit Radius 10, die entfernt wurde.

Kugel mit Radius 10 entfernt wurde, eine Fitting-Qualität wie in Abb. 51 gezeigt erreicht werden. Die mittlere Fitting-Qualität für Quader mit verschiedenen Volumina und verschiedenen Radien der herausgeschälten Kugeln ist in Abb. 52 dargestellt. Die maximale Rechenzeit wurde bei der Verwendung der vierten Momente (Bereich I) benötigt und betrug auf einem Pentium 4 Mobile mit 1.7 GHz 0.1s für das Fitting selbst, die Momentenberechnung dauerte zuvor für einen Quader mit Volumen 125000 hier 0.5s.

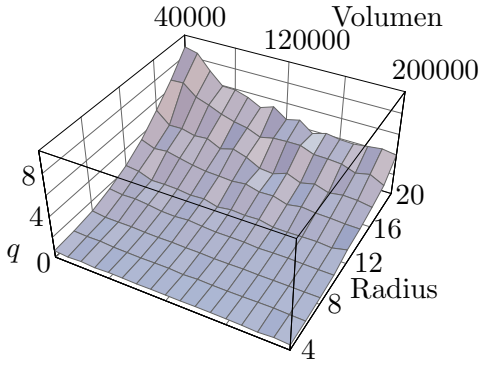


Abbildung 52: Mittlere Fitting-Qualität q für Quader mit gegebenem Volumen (von 40000 bis 200000 Voxel) und Kugeln mit gegebenem Radius (von 4 bis 20), die an einer Ecke zentriert und vom Objekt entfernt wurden. Für jede Volumen-Radius-Kombination wurden 50 Quader berechnet.

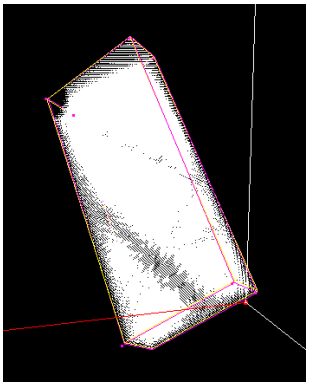


Abbildung 53: Ein Quader, aus dem eine an einer Ecke zentrierte Kugel herausgeschält wurde. Für dieses Beispiel wurde eine Fitting-Qualität q von 0.58 Voxeln erreicht.

9 Fitting von Superellipsoiden

Mit dem in Abschnitt 8 beschriebenen Algorithmus konnten sehr gute und robuste Fitting-Ergebnisse erzielt werden. Allerdings sind Quader bzw. Würfel eine doch recht eingeschränkte Art von Objekten, und es ist wünschenswert, für das Fitting auch andere Formen heranzuziehen.

Eine solche weitere interessante Klasse sind die Superellipsoide, die im nächsten Abschnitt vorgestellt werden.

9.1 Superellipsoide

Eine Ellipse in der Ebene kann durch die Parameterdarstellung

$$\mathbf{e}(u) = \begin{pmatrix} a\cos u \\ b\sin u \end{pmatrix} \quad \text{mit} \;\; u \in [-\pi, \pi)$$

beschrieben werden. Eine Verallgemeinerung der Ellipsen stellen die Superellipsen dar, die sich durch die Parameterdarstellung

$$\mathbf{s}(u) = \begin{pmatrix} a(\cos u)^{\varepsilon} \\ b(\sin u)^{\varepsilon} \end{pmatrix} \quad \text{mit} \;\; u \in [-\pi, \pi) \;\; \text{und} \;\; \varepsilon > 0$$

beschreiben lassen. Dabei kann über den Parameter ε die Form des Objekts beeinflusst werden. In Abb. 54 sind einige Beispiele für verschiedene Werte von ε angegeben.

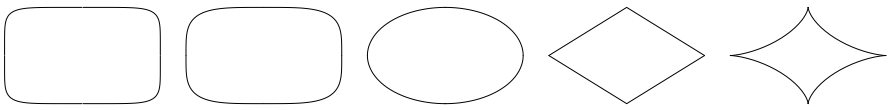


Abbildung 54: Verschieden geformte Superellipsen in Abhängigkeit vom Exponenten ε (Werte von links nach rechts 0.3, 0.5, 1.0, 2.0, 3.0).

Eine solche Erweiterung lässt sich natürlich auch im 3D-Raum vornehmen. Die Parameterdarstellung

$$\mathbf{e}(u, v) = \begin{pmatrix} a\cos u\cos v \\ b\cos u\sin v \\ c\sin u \end{pmatrix} \quad \text{mit} \;\; u \in \left[-\frac{\pi}{2}, \;\frac{\pi}{2}\right], \; v \in [-\pi, \pi)$$

kann zu

$$\mathbf{s}(u,v) = \begin{pmatrix} a(\cos u)^{\varepsilon_1}(\cos v)^{\varepsilon_2} \\ b(\cos u)^{\varepsilon_1}(\sin v)^{\varepsilon_2} \\ c(\sin u)^{\varepsilon_1} \end{pmatrix}$$

$$\text{mit } \; u \in \left[-\frac{\pi}{2}, \; \frac{\pi}{2}\right], \; v \in [-\pi, \pi), \; \varepsilon_1, \varepsilon_2 > 0,$$

verallgemeinert werden, wobei auch hier die Form des Objekts maßgeblich durch die beiden Parameter ε_1 und ε_2 bestimmt wird. Abb. 55 zeigt einige Beispiele dafür. Diese Darstellung ist in dieser Form oft in der Literatur zu finden, allerdings muss hier beachtet werden, dass die sin- und cos-Terme auch negative Werte annehmen können. Diese Fälle sollten durch Verwenden des absoluten Betrages dieser Terme und zusätzliche Betrachtung ihrer Vorzeichen behandelt werden.

Daneben lassen sich Superellipsoide auch durch die implizite Darstellung

$$\left(\left(\frac{x}{a}\right)^{\frac{2}{\varepsilon_2}} + \left(\frac{y}{b}\right)^{\frac{2}{\varepsilon_2}}\right)^{\frac{\varepsilon_2}{\varepsilon_1}} + \left(\frac{z}{c}\right)^{\frac{2}{\varepsilon_1}} = 1 \quad \text{mit } \varepsilon_1, \varepsilon_2 > 0$$

beschreiben.

Superellipsoide besitzen ein breites Anwendungsspektrum. So werden sie z. B. als Modellierungselemente in der Computergraphik eingesetzt. Weitere Anwendungen sind die Modellierung von Blasen in strömenden Flüssigkeiten sowie die Simulation von Partikeln in granulatartigen Materialien.

9.2 Fittingalgorithmus

In diesem Abschnitt wird ein Fitting-Algorithmus für Superellipsoide präsentiert, bei dem als Transformationen affine Abbildungen zugelassen sind. Damit vergrößert sich der Bereich möglicher anzupassender Formen erheblich auf „affin verzerrte“ Superellipsoide. Wie bereits bei den bisher beschriebenen Verfahren werden eine Separation der Transformationsmatrix sowie geeignete Normalisierungsbedingungen angegeben.

Es hat sich herausgestellt, dass eine affine Abbildung $\mathbf{A}$ im 3D-Raum analog zu der in Abschnitt 7.2 gezeigten Separation einer zweidimensionalen affinen Abbildung erfolgen kann. Wir betrachten dabei an dieser Stelle

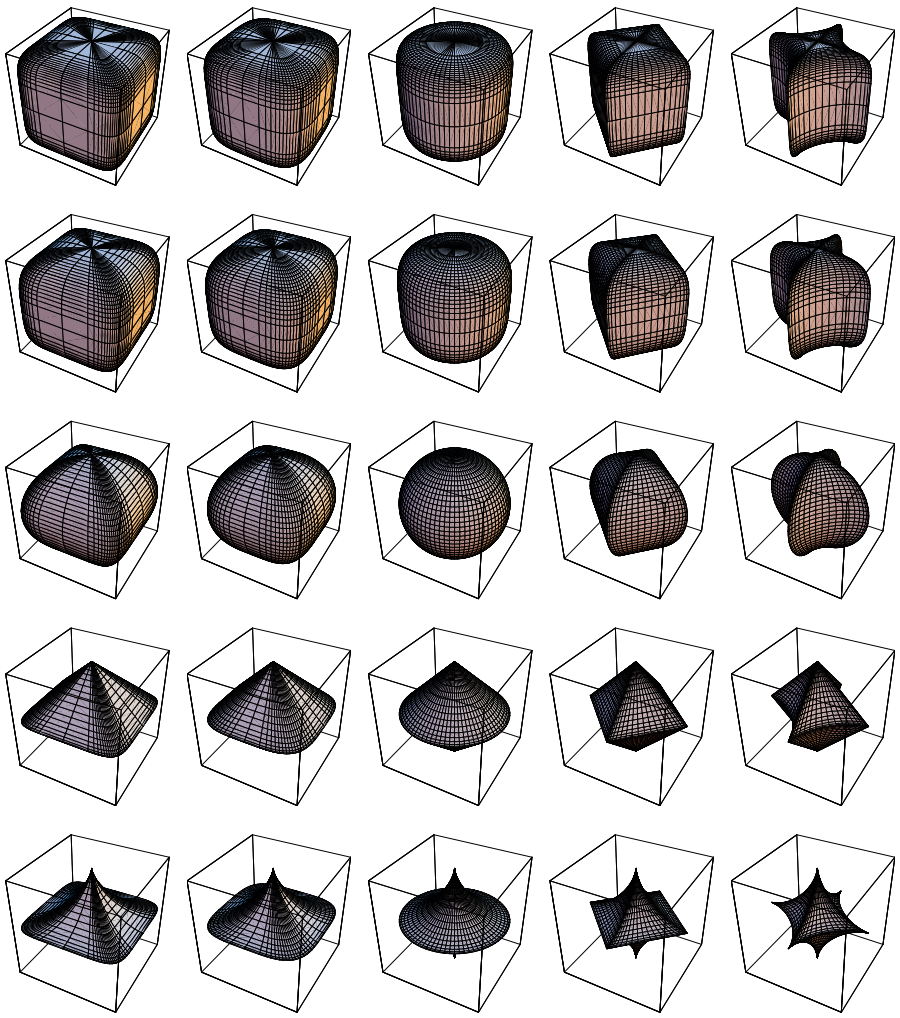


Abbildung 55: Verschiedene Superellipsoide, deren Form durch die Exponenten ε_1 und ε_2 bestimmt wird (0.3, 0.5, 1.0, 2.0, 3.0, ε_1 variiert zeilenweise, ε_2 spaltenweise).

allerdings nur Transformationen ohne Spiegelungsanteil. Dabei werden als Komponenten eine Rotation (die in drei einzelne Rotationen um die Koordinatenachsen zerlegt werden kann), eine anisotrope Skalierung sowie eine verallgemeinerte Scherung verwendet, so dass die Separation folgende Form aufweist:

$$\begin{aligned}
\mathbf{A} &= \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \\
&= R_Z \cdot R_Y \cdot R_X \cdot S \cdot D \\
&= \begin{pmatrix} \cos u & -\sin u & 0 \\ \sin u & \cos u & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos v & 0 & -\sin v \\ 0 & 1 & 0 \\ \sin v & 0 & \cos v \end{pmatrix} \cdot \ldots \\
&\quad \ldots \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos w & -\sin w \\ 0 & \sin w & \cos w \end{pmatrix} \begin{pmatrix} \delta & 0 & 0 \\ 0 & \varepsilon & 0 \\ 0 & 0 & \lambda \end{pmatrix} \begin{pmatrix} 1 & \alpha & \beta \\ 0 & 1 & \gamma \\ 0 & 0 & 1 \end{pmatrix} \\
&= \begin{pmatrix} | & | & | \\ C_1 & C_2 & C_3 \\ | & | & | \end{pmatrix}.
\end{aligned}$$

Um zu zeigen, dass diese Separation zulässig ist, wird die Berechnung der Parameter der Teiltransformationen aus den Parametern der Ausgangsmatrix $\mathbf{A}$ angegeben. Wir betrachten dabei zunächst die erste Spalte

$$C_1 = \begin{pmatrix} \delta \cos u \cos v \\ \delta \sin u \cos v \\ \delta \sin v \end{pmatrix} = \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \end{pmatrix}.$$

Durch Quadrieren aller drei Werte, Addition und Ausnutzen der Beziehung $\sin^2 x + \cos^2 x = 1$ sowie unter Berücksichtigung der Tatsache, dass hier nur reflektionsfreie Transformationen betrachtet werden sollen, erhält man

$$\delta = \sqrt{a_{11}^2 + a_{21}^2 + a_{31}^2}.$$

Ist δ bekannt, so kann mittels

$$\sin v = \frac{a_{31}}{\delta}$$

der Wert für v bestimmt werden, wobei hier genauer gesagt zwei Werte v_1 und v_2 infrage kommen. Für die beiden entsprechenden Werte für $\cos v_1$ und $\cos v_2$ gibt es jeweils genau einen passenden Wert u_1 bzw. u_2, da die Darstellung

$$\begin{pmatrix} (\delta \cos v) \cos u \\ (\delta \cos v) \sin u \end{pmatrix} = \begin{pmatrix} a_{11} \\ a_{21} \end{pmatrix}$$

als Polarkoordinatendarstellung eines Punktes in der Ebene interpretiert werden kann. Eine ähnliche Interpretation ist auch für die gesamte Spalte C_1 möglich, sie kann als Kugelkoordinatendarstellung eines Punktes im 3D-Raum aufgefasst werden. Ohne eine Einschränkung des Winkelbereichs ist diese Darstellung für u und v nicht eindeutig, für einen gegebenen Punkt kann neben u und v auch das Wertepaar $\pi + u$ und $\pi - v$ als Lösung verwendet werden, womit auch die Beziehung zwischen v_1 und v_2 geklärt wäre.

Der dritte Rotationsparameter w sowie die Parameter ε und α können aus der zweiten Spalte C_2 gewonnen werden:

$$C_2 = \begin{pmatrix} \alpha\delta \cos u \cos v - \varepsilon(\sin u \cos w + \cos u \sin v \sin w) \\ \alpha\delta \sin u \cos v + \varepsilon(\cos u \cos w - \sin u \sin v \sin w) \\ \alpha\delta \sin v + \varepsilon \cos v \sin w \end{pmatrix} = \begin{pmatrix} a_{12} \\ a_{22} \\ a_{32} \end{pmatrix}.$$

Hier erhält man durch die Substitution $x = \varepsilon \cos w$ und $y = \varepsilon \sin w$ das lineare Gleichungssystem

$$\begin{aligned} \alpha(\delta \cos u \cos v) + x(-\sin u) + y(-\cos u \sin v) &= a_{12} \\ \alpha(\delta \sin u \cos v) + x(\cos u) + y(-\sin u \sin v) &= a_{22} \\ \alpha(\delta \sin v) + y(\cos v) &= a_{32} \end{aligned}$$

mit den Determinanten

$$\begin{aligned} D &= \begin{vmatrix} \delta \cos u \cos v & -\sin u & -\cos u \sin v \\ \delta \sin u \cos v & \cos u & -\sin u \sin v \\ \delta \sin v & 0 & \cos v \end{vmatrix} \\ &= \delta \cos^2 u \cos^2 v + \delta \sin^2 u \sin^2 v + \delta \sin^2 u \cos^2 v + \delta \cos^2 u \sin^2 v \\ &= \delta, \end{aligned}$$

$$
\begin{aligned}
D_\alpha &= \begin{vmatrix} a_{12} & -\sin u & -\cos u \sin v \\ a_{22} & \cos u & -\sin u \sin v \\ a_{32} & 0 & \cos v \end{vmatrix} \\
&= a_{12} \cos u \cos v + a_{32} \sin^2 u \sin v + a_{22} \sin u \cos v + a_{32} \cos^2 u \sin v \\
&= a_{12} \cos u \cos v + a_{22} \sin u \cos v + a_{32} \sin v,
\end{aligned}
$$

$$
\begin{aligned}
D_x &= \begin{vmatrix} \delta \cos u \cos v & a_{12} & -\cos u \sin v \\ \delta \sin u \cos v & a_{22} & -\sin u \sin v \\ \delta \sin v & a_{32} & \cos v \end{vmatrix} \\
&= a_{22}\delta \cos u \cos^2 v - a_{12}\delta \sin u \sin^2 v - a_{32}\delta \sin u \cos u \sin v \cos v + \ldots \\
&\quad \ldots + a_{22}\delta \cos u \sin^2 v - a_{12}\delta \sin u \cos^2 v + a_{32}\delta \sin u \cos u \sin v \cos v \\
&= a_{22}\delta \cos u - a_{12}\delta \sin u,
\end{aligned}
$$

$$
\begin{aligned}
D_y &= \begin{vmatrix} \delta \cos u \cos v & -\sin u & a_{12} \\ \delta \sin u \cos v & \cos u & a_{22} \\ \delta \sin v & 0 & a_{32} \end{vmatrix} \\
&= a_{32}\delta \cos^2 u \cos v - a_{22}\delta \sin u \sin v + a_{32}\delta \sin^2 u \cos v - a_{12}\delta \cos u \sin v \\
&= a_{32}\delta \cos v - a_{12}\delta \cos u \sin v - a_{22}\delta \sin u \sin v.
\end{aligned}
$$

Verwendet man die Werte

$$
\begin{aligned}
u' &= \pi + u \\
v' &= \pi - v,
\end{aligned}
$$

so erhält man aufgrund der Beziehungen

$$
\begin{aligned}
\sin(\pi + u) &= -\sin u \\
\cos(\pi + u) &= -\cos u \\
\sin(\pi - v) &= \sin v \\
\cos(\pi - v) &= -\cos v
\end{aligned}
$$

für die entsprechenden Determinanten die Werte

$$
\begin{aligned}
D'_\alpha &= D_\alpha \\
D'_x &= -D_x \\
D'_y &= -D_y
\end{aligned}
$$

Damit ist das Ergebnis für α von der Wahl der Lösung unabhängig. Erhält man für die Werte u und v nach Auflösung der Substitution einen Wert w, so würde man für $\pi + u$ und $\pi - v$ als Lösung den Wert $\pi + w$ erhalten. Die beiden Matrizenprodukte

$$\begin{pmatrix} \cos u & -\sin u & 0 \\ \sin u & \cos u & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos v & 0 & -\sin v \\ 0 & 1 & 0 \\ \sin v & 0 & \cos v \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos w & -\sin w \\ 0 & \sin w & \cos w \end{pmatrix}$$

und

$$\begin{pmatrix} \cos(\pi+u) & -\sin(\pi+u) & 0 \\ \sin(\pi+u) & \cos(\pi+u) & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos(\pi-v) & 0 & -\sin(\pi-v) \\ 0 & 1 & 0 \\ \sin(\pi-v) & 0 & \cos(\pi-v) \end{pmatrix} \cdot \ldots$$

$$\cdot \ldots \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(\pi+w) & -\sin(\pi+w) \\ 0 & \sin(\pi+w) & \cos(\pi+w) \end{pmatrix}$$

liefern das identische Ergebnis

$$\begin{pmatrix} \cos u \cos v & -\cos w \sin u + \cos u \sin v \sin w & \cos u \cos w \sin v + \sin u \sin w \\ \cos v \sin u & \cos u \cos w + \sin u \sin v \sin w & \cos w \sin u \sin v - \cos u \sin w \\ \sin v & \cos v \sin w & \cos v \cos w \end{pmatrix},$$

so dass sich die Mehrdeutigkeit hier ausschließlich auf den Rotationsanteil beschränkt und nur zutage tritt, falls die Zerlegung in die drei Rotationen um die Koordinatenachsen betrachtet wird.

Wie schon kurz erwähnt, erhält man die Werte für ε und w bzw. w' durch Auflösen der oben angegebenen Substitution.

Die restlichen Parameter β, γ und λ können anschließend aus der dritten Spalte gewonnen werden:

$$C_3 = \begin{pmatrix} \beta\delta \cos u \cos v - \varepsilon\gamma(\sin u \cos w + \cos u \sin v \sin w) \\ \qquad +\lambda(\sin u \sin w - \cos u \sin v \cos w) \\ \beta\delta \sin u \cos v + \varepsilon\gamma(\cos u \cos w - \sin u \sin v \sin w) \\ \qquad -\lambda(\cos u \sin w + \sin u \sin v \cos w)) \\ \beta\delta \sin v + \varepsilon\gamma \cos v \sin w + \lambda \cos v \cos w \end{pmatrix} = \begin{pmatrix} a_{13} \\ a_{23} \\ a_{33} \end{pmatrix}.$$

Sie stellt einfach ein lineares Gleichungssystem mit den gesuchten Parametern als Unbekannten dar, und man kann sich leicht davon überzeugen,

dass die Koeffizienten von der Wahl der Rotationsparameter (u, v, w bzw. u', v', w') nicht beeinflusst werden.

Damit ist gezeigt, dass die oben angegebene Separation der affinen Transformation zulässig ist, unter Berücksichtigung der Zerlegung der Rotation in drei Einzelrotationen um die Koordinatenachsen gibt es dabei genau zwei verschiedene Parametersätze $(u, v, w, \delta, \varepsilon, \lambda, \alpha, \beta, \gamma)$ und $(u', v', w', \delta, \varepsilon, \lambda, \alpha, \beta, \gamma)$.

Unter Verwendung dieser Separation lässt sich nun ein Verfahren zur Normalisierung mit entsprechenden Bedingungen angeben. Im ersten Schritt betrachten wir dabei die Scherungsmatrix. Allgemein transformieren sich dabei die Momente auf folgende Weise:

$$m'_{pqr} = \iiint_{object} (x + \alpha y + \beta z)^p (y + \gamma z)^q z^r \, dx \, dy \, dz.$$

Analog zum zweidimensionalen Fall betrachten wir dabei insbesondere die Momente m_{110}, m_{101}, m_{011}:

$$\begin{aligned}
m'_{110} &= \iiint_{object} (x + \alpha y + \beta z)(y + \gamma z) dx \, dy \, dz \\
&= \iiint_{object} (xy + \gamma xz + \alpha y^2 + \alpha\gamma yz + \beta zy + \beta\gamma z^2) dx \, dy \, dz \\
&= m_{110} + \gamma m_{101} + \alpha m_{020} + (\alpha\gamma + \beta) m_{011} + \beta\gamma m_{002}
\end{aligned}$$

$$\begin{aligned}
m'_{101} &= \iiint_{object} (x + \alpha y + \beta z) z dx \, dy \, dz \\
&= m_{101} + \alpha m_{011} + \beta m_{002}
\end{aligned}$$

$$\begin{aligned}
m'_{011} &= \iiint_{object} (y + \gamma z) z dx \, dy \, dz \\
&= m_{011} + \gamma m_{002}.
\end{aligned}$$

Bei Vorgabe der Bedingungen

$$m'_{110} = 0, \quad m'_{101} = 0, \quad m'_{011} = 0 \tag{2}$$

können die Scherungsparameter auf folgende Weise bestimmt werden:

$$\begin{aligned}\alpha &= \frac{m_{101}m_{011} - m_{110}m_{002}}{m_{020}m_{002} - m_{011}^2},\\ \beta &= \frac{m_{110}m_{011} - m_{101}m_{020}}{m_{020}m_{002} - m_{011}^2},\\ \gamma &= -\frac{m_{011}}{m_{002}}.\end{aligned}$$

Im zweiten Schritt erfolgt die Normalisierung der anisotropen Skalierung. Durch sie werden die Momente gemäß

$$\begin{aligned}m''_{pqr} &= \begin{vmatrix}\delta & 0 & 0\\ 0 & \varepsilon & 0\\ 0 & 0 & \lambda\end{vmatrix} \cdot \iiint_{object} (\delta x)^p (\varepsilon y)^q (\lambda z)^r \, dx\, dy\, dz\\ &= \delta^{p+1}\varepsilon^{q+1}\lambda^{r+1} m'_{pqr}\end{aligned}$$

transformiert, insbesondere gilt

$$m''_{200} = \delta^3\varepsilon\lambda m'_{200}, \quad (3)$$
$$m''_{020} = \delta\varepsilon^3\lambda m'_{020}, \quad (4)$$
$$m''_{002} = \delta\varepsilon\lambda^3 m'_{002}. \quad (5)$$

Die bisherigen Normalisierungsbedingungen (2) werden dadurch nicht verletzt.

Als Bedingungen geben wir hier

$$m''_{200} = 1, \quad m''_{020} = 1, \quad m''_{002} = 1$$

vor. Aus Gleichung (5) erhält man dann

$$\delta = \frac{1}{\varepsilon\lambda^3 m'_{002}}$$

und damit aus (3) bzw. (4)

$$\frac{m'_{200}}{\varepsilon^2\lambda^8 m'^3_{002}} = 1 \quad \text{bzw.} \quad \frac{\varepsilon^2 m'_{020}}{\lambda^2 m'_{002}}.$$

Schließlich folgt daraus

$$\begin{aligned}\lambda &= \sqrt[10]{\frac{m'_{200}m'_{020}}{m'^4_{002}}},\\ \delta &= \sqrt[10]{\frac{m'_{020}m'_{002}}{m'^4_{200}}},\\ \varepsilon &= \sqrt[10]{\frac{m'_{002}m'_{200}}{m'^4_{020}}}.\end{aligned}$$

Der dritte Schritt ist die Normalisierung der Rotation. Für die oben beschriebene Separation und Normalisierung im zweidimensionalen Fall war es dabei möglich, den Rotationswinkel φ direkt zu bestimmen, indem die Beziehung

$$m'_{30} + m'_{12} = (m_{30} + m_{12})\cos\varphi - (m_{03} + m_{21})\sin\varphi = 0$$

verwendet wurde.

Für den dreidimensionalen Fall konnte eine solche Lösung bisher leider nicht gefunden werden. Um dennoch eine Normalisierung der Rotation zu erreichen, hat es sich in Experimenten als günstig erwiesen, die Bedingungen

$$\begin{aligned} m'''_{310} = 0, \quad m'''_{301} = 0\\ m'''_{130} = 0, \quad m'''_{031} = 0\\ m'''_{103} = 0, \quad m'''_{013} = 0\end{aligned}$$

zu verwenden und eine Optimierung

$$\begin{aligned}&(m'''_{310}(u,v,w))^2 + (m'''_{301}(u,v,w))^2 + (m'''_{130}(u,v,w))^2 + \ldots\\ &\ldots + (m'''_{031}(u,v,w))^2 + (m'''_{103}(u,v,w))^2 + (m'''_{013}(u,v,w))^2 \quad\longrightarrow\quad \text{minimum}\end{aligned}$$

vorzunehmen, um die drei Rotationswinkel $u, v, w \in [0, 2\pi)$ numerisch zu bestimmen, mit denen das Objekt letztendlich in die Standardlage gebracht wird.

Daran anschließend können nun die Parameter bestimmt werden, die dasjenige Superellipsoid definieren, welches das normalisierte Objekt optimal beschreibt. Dazu werden die Momente

$$m'''_{000}, \quad m'''_{400}, \quad m'''_{040}, \quad m'''_{004}, \quad m'''_{220}, \quad m'''_{202}, \quad m'''_{022}$$

verwendet. Ihre theoretischen Werte berechnen sich für gegebene Parameter a, b, c, ε_1, ε_2 gemäß

$$\begin{aligned}\mu_{pqr}(a,b,c,\varepsilon_1,\varepsilon_2) &= \frac{2}{p+q+2}a^{p+1}b^{q+1}c^{r+1}\varepsilon_1\varepsilon_2\cdot\ldots \\ &\quad \ldots\cdot B\left((r+1)\frac{\varepsilon_1}{2},(p+q+2)\frac{\varepsilon_1}{2}+1\right)\cdot\ldots \\ &\quad \ldots\cdot B\left((q+1)\frac{\varepsilon_2}{2},(p+1)\frac{\varepsilon_2}{2}\right),\end{aligned}$$

da alle Indizes der Momente geradzahlig sind, Momente mit ungeradzahligen Indizes haben den Wert 0. $B(x,y)$ ist dabei die Beta-Funktion, die durch

$$B(x,y)=\frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$$

definiert ist. Details zur Herleitung der Momentenberechnung sind in [13] zu finden.

Unter Benutzung dieser Beziehung kann nun eine Optimierung

$$\begin{aligned}&\left(m'''_{000}-\mu_{000}(a,b,c,\varepsilon_1,\varepsilon_2)\right)^2+\ldots \\ &\ldots+\left(m'''_{400}-\mu_{400}(a,b,c,\varepsilon_1,\varepsilon_2)\right)^2+\left(m'''_{040}-\mu_{040}(a,b,c,\varepsilon_1,\varepsilon_2)\right)^2+\ldots \\ &\ldots+\left(m'''_{004}-\mu_{004}(a,b,c,\varepsilon_1,\varepsilon_2)\right)^2+\left(m'''_{220}-\mu_{220}(a,b,c,\varepsilon_1,\varepsilon_2)\right)^2+\ldots \\ &\ldots+\left(m'''_{202}-\mu_{202}(a,b,c,\varepsilon_1,\varepsilon_2)\right)^2+\left(m'''_{022}-\mu_{022}(a,b,c,\varepsilon_1,\varepsilon_2)\right)^2 \\ &\longrightarrow \text{minimum}\end{aligned}$$

mit $a,b,c,\varepsilon_1,\varepsilon_2>0$ vorgenommen werden, um den optimalen Parametersatz zu bestimmen.

Zu beachten ist dabei jedoch, dass die oben angegebene Berechnung der Momente für Superellipsoide gilt, die sich in der speziellen durch die Parameterdarstellung in Abschnitt 9.1 definierten Lage (ausgerichtet entlang der z-Achse) befinden. Allerdings kann durch die Normalisierungsbedingungen für die Rotation nicht sichergestellt werden, dass auch genau diese Lage erreicht wird. Die Bedingungen sind erfüllt, wenn (aber nicht genau dann wenn) das Objekt entlang einer Koordinatenachse ausgerichtet ist und auch wenn es zusätzlich noch um $\pi/4$ um diese Achse gedreht ist. Daher muss die Parameteroptimierung für jede dieser sechs Lagen durchgeführt werden, und das beste Resultat wird als Parametersatz verwendet.

Darüber hinaus gibt es offenbar gemäß den durchgeführten Experimenten für nichtkonvexe Objekte (d. h. solche, bei denen die resultierenden Exponenten ε_1 und ε_2 größer als 2 sind) noch spezielle Lagen, in denen ebenfalls die Bedingungen für die Rotationsnormalisierung erfüllt sind. In diesen Fällen erhält man aber durch die Optimierung kein sinnvolles Ergebnis für die Parameter, so dass im praktischen Einsatz diese Fälle leicht verworfen werden können. In den durchgeführten Experimenten war es stets möglich, durch Wahl mehrerer Startwerte ein korrektes Ergebnis zu erhalten.

Durch die Anwendung der Inversen der Normalisierungstransformation auf das Superellipsoid in Standardlage mit dem optimalen Parametersatz kann dann letztlich das Objekt erhalten werden, dass die vorgegebene Voxelmenge am besten annähert.

9.3 Experimentelle Ergebnisse

Um die Qualität des Verfahrens zu beurteilen, wurden innerhalb eines Voxelwürfels der Größe 200^3 zufällig affin verzerrte Superellipsoide generiert. Zur Bestimmung eines Qualitätsmaßes wurden einerseits die Randvoxel der gegebenen Voxelmenge ermittelt, andererseits wurde der Rand des Fittingergebnisses erzeugt. Für jedes Voxel der einen Randmenge wurde dann das nächstgelegene Voxel der jeweils anderen Randmenge gesucht und der Abstand zwischen beiden bestimmt. Das Maximum der Abstände all dieser Randvoxelpaare wurde dann als Maß für die Fitting-Qualität herangezogen.

Das angegebene Verfahren liefert dabei sehr gute Fittingergebnisse. So wurde für 300 zufällig generierte affin transformierte Superellipsoide eine durchschnittliche Fitting-Qualität von 1.5 erhalten. Die mittlere Rechenzeit betrug auf einem Pentium 4 Mobile mit 1.7 GHz 2.8s (0.6s Momentenberechnung, 2.2s Fitting).

Selbst wenn die Form des Objekts durch größere fehlende oder hinzugefügte Teile modifiziert wurde, waren die Abweichungen nur gering, falls die zu verwendenden Exponenten ε_1 und ε_2 im voraus bekannt waren. Beispiele dafür sind in Abb. 56 gezeigt. Im linken und mittleren Bild wurde ein Superellipsoid mit den Parametern $a = 40$, $b = 60$, $c = 80$, $\varepsilon_1 = 1$ und $\varepsilon_2 = 2$ benutzt, und es wurde jeweils eine Kugel mit dem Radius 30 entfernt bzw. hinzugefügt. Als Fitting-Qualität wurden dabei die Werte 3.16 bzw.

Abbildung 56: Drei Superellipsoide (grau) mit größeren entfernten oder hinzugefügten Kugeln und das erhaltene Fittingergebnis (weiß). Erläuterung siehe Abschnitt 9.3.

8.25 erhalten. Im rechten Bild wurde für die Exponenten der Wert 2.5 verwendet, und der Radius der vom Objekt entfernten Kugel beträgt 20, was eine Fitting-Qualität von 3.61 liefert.

10 Zusammenfassung

Im zweiten Teil der Arbeit wurden nach einer kurzen Erläuterung des Grundprinzips der Normalisierung und einem Beispiel für eine Separation einer affinen Transformation in 2D zwei neue Algorithmen für das Fitting von Quadern bzw. Würfeln sowie von Superellipsoiden vorgestellt. Für Quader und Würfel wurden insbesondere auftretende Entartungen des Trägheitsellipsoids für die Normalisierung der Rotation erfolgreich durch das Einbeziehen der Momente vierter Ordnung behandelt. Um das Fitting für die Superellipsoide zu realisieren, wurde eine Separation einer affinen Transformation im 3D-Raum in eine Rotation, eine anisotrope Skalierung und eine verallgemeinerte Scherung vorgenommen. Für beide Objektklassen wurden dabei sehr gute und auch größeren Störungen gegenüber robuste Fitting-Qualitäten erzielt.

Diese beiden Verfahren können ein guter Ausgangspunkt für weiterführende Arbeiten im Bereich 3D-Fitting mittels Normalisierung sein. So kann es trotz der großen Formenvielfalt, die durch die affin-transformierten Superellipsoide abgedeckt wird, sinnvoll sein, auch noch für weitere Objektklassen wie Kugeln, Ellipsoide, oder Parallelepipede eigenständige Fittingroutinen zu entwickeln. Damit würden auch dafür die Vorteile des Fittings per Normalisierung nutzbar gemacht, die vor allem darin bestehen, dass man einfach mit der Beschreibung durch eine Pixel- bzw. hier Voxelmenge auskommt, keine Objekte suchen und beschreiben muss, die den Rand des Objekts charakterisieren und man daneben noch eine deutliche Verkleinerung des Suchraums für die freien Parameter erhält.

Literatur

[1] Dimri, J.; Gurumoorthy, B.: *Handling Sectional Views in Volume-Based Approach to Automatically Construct 3D Solid from 2D Views.* Computer Aided Design, 37(5), pp. 485-495, 2005.

[2] Ditrich, F.; Suesse, H.; Voss, K.: *Vectorization-Free Reconstruction of 3D CAD Models from Paper Drawings.* 26. DAGM-Symposium, Tübingen, Germany, pp. 463-470, 2004.

[3] Ditrich, F.; Suesse, H.; Voss, K.: *Vectorization-Free Reconstruction of 3D CAD Models from Paper Drawings.* Image Analysis and Recognition: International Conference, ICIAR 2004, Porto, Portugal, pp. 629-637, 2004.

[4] Ditrich, F.; Suesse, H.; Voss, K.: *A New Efficient Algorithm for Fitting Rectangular Boxes and Cubes in 3D.* International Conference on Image Processing, ICIP 2005, Genoa, Italy, pp. 1134-1137, 2005.

[5] Ditrich, F.; Suesse, H.: *Robust Fitting of 3D Objects Using Normalization.* Wird zur Veröffentlichung eingereicht.

[6] Ditrich, F.; Suesse, H.: *A System for Reconstruction of 3D CAD Models from Paper Drawings.* Wird zur Veröffentlichung eingereicht.

[7] Foley, J. D.; van Dam, A.; Feiner, S. K.; Hughes, J. F.: *Computer Graphics: Principles and Practice, Second Edition.* Addison-Wesley, 1990.

[8] Ganesan, R.; Devarajan, V.: *Intersecting features extraction from 2D orthographic projections.* Computer Aided Design 30(11), pp. 863-873, 1998.

[9] Galvez, J. M.; Canton, M.: *Normalization and Shape Recognition of Three-Dimensional Objects by 3D Moments.* Pattern Recognition, 26(5), pp. 667-681, 1993.

[10] Geschke, H. W.; Helmetag, M.; Wehr, W.; Böttcher, P.; Forberg, R.: *Technisches Zeichnen.* 23. neubearb. und erw. Aufl.; B. G. Teubner, Stuttgart, Leipzig; Beuth Verlag, Berlin, Wien, Zürich, 1998.

[11] Graham, R. L.; Knuth, D. E.; Patashnik, O.: *Concrete Mathematics.* Addison-Wesley, 1994.

[12] Hough, P. V. C.: *Method and Means for Recognizing Complex Patterns.* US Patent 3069654 (1962).

[13] Jaklič, A.; Solina, F.: *Moments of Superellipsoids and their Application to Range Image Registration.* IEEE Trans. Systems, Man and Cybernetics, Part B 33(4), 2003, pp. 648-657.

[14] Kuo, M. H.: *Reconstruction of Quadric Surface Solids from Three-View Engineering Drawings.* Computer Aided Design, 30(7), pp. 517-527, 1998.

[15] Meeran, S.; Taib, J. M.: *A Generic Approach to Recognising Isolated, Nested and Interacting Features from 2D Drawings.* Computer Aided Design 31(4), pp. 891-910, 1999.

[16] Piegl, L. A.: *Ten Challenges in Computer-Aided Design.* Computer Aided Design, 37(4), pp. 461-470, 2005.

[17] Piegl, L. A.; Tiller, W.: *The NURBS Book.* Springer-Verlag, Berlin, Heidelberg, 1997.

[18] Shum, S. S. P.; Lau, W. S.; Yuen, M. M. F.; Yu, K. M.: *Solid reconstruction from orthographic views using 2-stage extrusion.* Computer Aided Design 33(1), pp. 91-102, 2001.

[19] Suesse, H.; Ditrich, F.: *Robust Determination of Rotation-Angles for Closed Regions Using Moments.* International Conference on Image Processing, ICIP 2005, Genoa, Italy, pp. 337-340, 2005.

[20] Tanaka, M.; Anthony, L.; Kaneeda, T.; Hirooka, J.: *A Single Solution Method for Converting 2D Assembly Drawings to 3D Part Drawings.* Computer Aided Design 36(8), pp. 723-734, 2004.

[21] Voss, K.; Suesse, H.: *Adaptive Modelle und Invarianten für zweidimensionale Bilder.* Verlag Shaker, Aachen, 1995.

[22] Voss, K.; Suesse, H.: *Invariant Fitting of Planar Objects by Primitives.* IEEE Trans. PAMI, 19, pp. 80-83, 1997.

[23] Voss, K.; Suesse, H.: *A New One-Parametric Fitting Method for Planar Objects.* IEEE Trans. PAMI, 21, pp. 646-651, 1999.

[24] Voss, K.; Suesse, H.: *A New Efficient Algorithm for Fitting of Rectangles and Squares.* International Conference on Image Processing, ICIP 2001, Barcelona, Spain, pp. 809-812, 2001.

[25] Wang, K. P. W.: *Affin Invariant Moment Method of Three-Dimensional Object Identification.* Ph. D. Thesis. Syracuse University, 1977.

[26] Wesley, M. A.; Markowsky, G.: *Fleshing Out Wire Frames.* IBM Journal of Research and Development, 24(5), pp. 582-597, September 1980.

[27] Wesley, M. A.; Markowsky, G.: *Fleshing Out Projections.* IBM Journal of Research and Development, 25(6), pp. 934-954, November 1981.

[28] Xu, L.; Oja, E: *Randomized Hough Transform (RHT): Basic Mechanisms, Algorithms, and Computational Complexities.* CVGIP: Image Understanding, 57, pp. 131-154, 1993.

[29] Yan, Q.-W.; Philip Chen, C. L.; Tang, Z.: *Efficient algorithm for the reconstruction of 3D objects from orthographic projections.* Computer Aided Design, 26(9), pp. 699-717, September 1994.

[30] ISO 10303-214:2003, Industrial automation systems and integration – Product data representation and exchange – Part 214: Application protocol: Core data for automotive mechanical design processes.

[31] ISO 10303-224:2001, Industrial automation systems and integration – Product data representation and exchange – Part 224: Application protocol: Mechanical product definition for process planning using machining features.

[32] ISO 14649-10:2004, Industrial automation systems and integration – Physical device control – Data model for computerized numerical controllers – Part 10: General process data.

[33] ISO 14649-11:2004, Industrial automation systems and integration – Physical device control – Data model for computerized numerical controllers – Part 11: Process data for milling.

[34] ISO/DIS 14649-12:2004, Industrial automation systems and integration – Physical device control – Data model for computerized numerical controllers – Part 12: Process data for turning.

[35] http://www.solidworks.de

[36] http://www.isdcad.com/de

[37] http://www.graphisoft.de

[38] http://www.acado.ch

[39] http://www.autodesk.de

[40] http://www.solidedge.com

[41] http://www.mastercam.de

[42] http://www.ptc.de

[43] http://www.openmind-tech.de

[44] http://www.ntcadcam.co.uk

[45] http://www.geovision.de

[46] http://www.honeywell.com

[47] http://www.opencascade.org

Lebenslauf

Name:	Ditrich, Frank
Geburtsdatum:	23.11.1971
Geburtsort:	Gera

1978-1986	8. Polytechnische Oberschule Gera
1986-1990	Spezialschule mathematisch-naturwissenschaftlich-technischer Richtung „Carl Zeiss“ Jena
1990	Abitur
1990-11/1995	Studium der Mathematik, Friedrich-Schiller-Universität Jena
11/1995	Abschluss als Diplom-Mathematiker
12/1995-12/1996	Zivildienst, Volkssolidarität e. V. Gera
11/1996-9/1998	wissenschaftlicher Mitarbeiter im Drittmittelprojekt „Implementierung einer Virtual-Reality-Betriebssystemoberfläche“, Friedrich-Schiller-Universität Jena, Institut für Informatik
seit 1/1999	Mitarbeiter für Vorlaufentwicklung, DAKO GmbH Jena
seit 4/2003	Promotionsstudent, Friedrich-Schiller-Universität Jena, Institut für Informatik

Jena, d. 20.03.2007

Ehrenwörtliche Erklärung

Hiermit erkläre ich,

- dass mir die Promotionsordnung der Fakultät bekannt ist,
- dass ich die Dissertation selbst angefertigt und alle von mir benutzten Hilfsmittel, persönlichen Mitteilungen sowie Quellen in meiner Arbeit angegeben habe,
- dass ich die Hilfe eines Promotionsberaters nicht in Anspruch genommen habe und dass Dritte weder unmittelbar noch mittelbar geldwerte Leistungen von mir für Arbeiten erhalten haben, die im Zusammenhang mit dem Inhalt der vorgelegten Dissertation stehen,
- dass ich die Dissertation noch nicht als Prüfungsarbeit für eine staatliche oder andere wissenschaftliche Prüfung eingereicht habe,
- dass ich bei keiner anderen Hochschule eine Abhandlung als Dissertation eingereicht habe.

Jena, den 20.03.2007